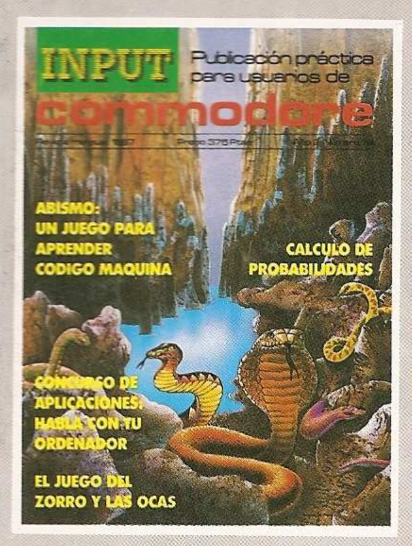


INPUT
Publicación práctica para usuarios de
commodore
Revista mensual 1987
Precio 375 Ptas
Año 2 Número 19
ABISMO: UN JUEGO PARA APRENDER CODIGO MAQUINA
CALCULO DE PROBABILIDADES
CONCURSO DE APLICACIONES: HABLA CON TU ORDENADOR
EL JUEGO DEL ZORRO Y LAS OCAS



AÑO 2 NUMERO 19

DIRECTOR: Manuel Pérez
DIRECTOR DE ARTE: Luis F. Balaguer
REALIZACION GRAFICA: Didac Tudela
COLABORADORES: Antonio Pliego, Xavier Ferrer, Josep M.ª Gils, Christophe Pais, Jaime Mardones, Equipo Molisoft, Carles Bartolomé, Ramón Ollé, Angels Alvarez
FOTOGRAFIA: Ernesto Walfisch, Joan Boada

INPUT Commodore es una publicación de PLANETA-DE AGOSTINI, S.A.
GERENTE DIVISION DE REVISTAS:
Sebastián Martínez

PUBLICIDAD: José Real-Grupo Jota
Madrid: c/ General Varela, 35
Teléf. 270 47 02/03
Barcelona: Avda. de Sarriá, 11-13, 1.°
Teléf. 250 23 99

FOTOMECANICA: TECFA, S.A.

IMPRESION: Sirven Grafic
c/ Gran Vía, 754-756, 08013 Barcelona
Depósito legal: B. 38.114-1986

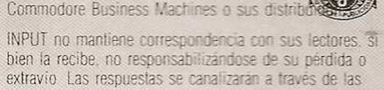
SUSCRIPCIONES: EDISA
López de Hoyos, 141. 28002 Madrid
Teléf. (91) 415 97 12

REDACCION:
Aribau, 185, 1.°
08021 Barcelona

DISTRIBUIDORA:
R.B.A. PROMOTORA DE EDICIONES, S.A.
Calle B, n.° 11. Sector B. Zona Franca
08004 Barcelona

El precio será el mismo para Canarias que para la Península y en él irá incluida la sobretasa aérea.

INPUT Commodore es una publicación controlada por



INPUT Commodore es independiente y no está [illegible]da a Commodore Business Machines o sus distribu[illegible]

INPUT no mantiene correspondencia con sus lectores, si bien la recibe, no responsabilizándose de su pérdida o extravío. Las respuestas se canalizarán a través de las secciones adecuadas en estas páginas.

INPUT commodore

SUMARIO

EDITORIAL

NUEVAS NECESIDADES DE LOS USUARIOS

A medida que pasa el tiempo y los usuarios conocen mejor los equipos con los que trabajan, sus demandas hacia los medios de comunicación especializados, también se modifican.

INPUT no podía permanecer insensible a esas nuevas tendencias, tan naturales como el paso del tiempo, y decidió ir renovándose a sí misma.

Ya habréis observado que tanto en algunos contenidos como en las presentaciones, en números anteriores, se han introducido novedades. En este ejemplar que ahora tenéis en vuestras manos los cambios ya afectan a una parte considerable de la revista.

De entrada debemos aclarar que en ningún caso se pretende romper con una tradición que ha conectado muy bien con los deseos de nuestros lectores, como nos consta por la aceptación y venta de INPUT y por las numerosas cartas que en la redacción se reciben todos los meses.

Sí se anhela, en cambio, dar cabida a esas nuevas necesidades que antes mencionábamos. No es un reto fácil, pues se trata de intentar conciliar cosas aparentemente tan contradictorias como la información más completa sobre videojuegos y todo el entorno que ello comporta, por un lado, y el diseño y presentación de utilidades, aplicaciones y nociones de programación de forma educativa, clara, comprensible y amena, por otro.

Esta necesidad permanente de formación se inscribe en el marco de una oferta informática que siempre ofrece más a un menor coste. Lógicamente, quienes hoy comienzan con pequeños micros domésticos operarán mañana con potentes ordenadores personales. En esta perspectiva el poso de conocimientos generado en la primera fase será de incalculable utilidad en el futuro.

Así pues, el mundo de la informática, y aunque con rasgos propios la microinformática doméstica forma parte de él, tiene que construir una nueva forma de hacer que forzosamente pasa por conseguir que su presencia sea imprescindible tanto a la hora de trabajar como a la de jugar. En este empeño INPUT continúa a vuestro servicio pensando siempre en ofreceros cosas nuevas a la vez que prácticas.

Vosotros sois quienes finalmente decidiréis si, con tales fundamentos, INPUT responde plenamente a vuestras demandas.

AMPLIA TU CURSO DE MECANOGRAFIA

- UN PASO MAS
- ADICION DE LOS NUMEROS Y EL RESTO DEL TECLADO
- MAS SIMBOLOS CON LA TECLA SHIFT

Esta vez puedes extender tu pericia a las teclas de números y a todos los restantes símbolos del teclado. Hay además un programa de juego que te ayudará a aumentar todavía más tu rapidez en la escritura.

Si has trabajado ya con el instructor mecanográfico que publicamos en INPUT, estarás familiarizado con todas las teclas de letras y de puntuación de las tres hileras inferiores del teclado. Pero aún no has tocado las siempre importantes (en los listados de programas) teclas numéricas, ni has aprendido cómo modificar las teclas para obtener letras mayúsculas o minúsculas, así como los símbolos adicionales y de puntuación disponibles.

El programa de esta sección te muestra cómo añadir todo esto a tus crecientes habilidades mecanográficas. Además, hay un ejercicio que te permitirá incrementar tu velocidad, tu seguridad y tu ritmo. Y además es divertido.

ADICION DE LOS NUMEROS

Al igual que con el programa anterior la adición de la línea de números del teclado puede llevarse a cabo con una simple modificación del programa original utilizado en las lecciones anteriores. Para ello, carga el último programa y teclea estas líneas adicionales. Algunas de ellas sustituyen a líneas existentes, mientras que otras van numeradas de modo que quedarán incluidas entre las que ya tienes.

```
10 DIM W$(21), WO$(28): FOR
   Z=1 TO 28: READ WO$(Z):
   NEXT
20 FOR Z=1 TO 40: LI$=LI$+"-
   "NEXT :LI$="
   [CTRL+2]"+LI$
30 PRINT"[SHIFT+CLR/HOME]
   [CTRL+8]"TAB(7)"*****ME
   CANOGRAFIA*****"
40 A$="1A2S3D4F5G6H7J8K
   9L0: ":POKE 54296, F515
   :GOTO380
50 TI$="000000":S=0:WW=1
60 N=0:POKE198,0
70 IFK=5ANDP>0THENN=N+1
   :P=P+1:GOTO140
80 PRINT"[CLR/HOME][8*CRSR
   ABAJO]":IFK<3THENPRINT
   TAB(10)
   "[CTRL+4][CTRL+9]"
   A$POKE198,0
90 X=INT(RND(1)*20)+1
   :N=N+1:P=P+1:IFK=1
   THENX=N:GOTO120
100 IFK=3THENX=INT
   (RND(1)*20)+1:PRINT TAB
   (18)"[CTRL+4][COMM+A]-
   [COMM+S][CRSRABAJO][
   3*CRSRIZQDA.]I[CTRL+6]"
   MID$(A$,X,1)
   "[CTRL+4][CRSRABAJO][
   3*CRSRIZQDA.][COMM+Z]-
   [COMM+X]"
110 IFK=4THENPRINT
   TAB(16)"[CTRL+4][12
   ESPACIOS][12*CRSRIZQDA.
   ]"W$(WW):X=N+6
120 IFK<3ORK=4THENPRINT
   "[CLR/HOME][7*CRSR
   ABAJO]"TAB(10+(X-
   1))"[CTRL+2]*"
130 IFK=5THENFORZ=1TO
   5:PRINT"[CTRL+8]"W$(Z);
   :NEXTZ:PRINT:PRINT
   "[CTRL+9][CTRL+4";
140 GETK$:IFK$=""THEN140
150 IFK>3AND
   K$MID$(W$(WW),N1)
   THEN210
160 IFK>3THEN180
170 IFK$=MID$(A$,X,1)
   THEN210
180 W=54276:A=54277
190 POKEW,33:POKEA,50:POKE
   54273,30:S=S+1
200 POKE54273,0:POKEW,
   32:GOTO140
210 W=54276:A=54277
220 IFK<3ORK=4THENPRINT
   "[CLR/HOME][7*CRSR
   ABAJO]"TAB(10+(X-1))""
230 IFK=5THENPRINTK$;
240 POKEW,33:POKEA,
   50:POKE54273,150
250 POKE54273,0:POKEW,32
260 IFK=1ANDN<20THEN70
270 IFK>3ANDN<LEN(W$(WW))
   THEN70
280 IF(K<4ANDK>1)ANDN<20
   THEN70
290 IFK>3THENNEXTWW
300 WW=VAL(MID$(TI$,3,
```



```
2))*6Ø+VAL(RIGHT$(TI$,2))
31Ø PRINT
    "[CTRL+Ø][SHIFT+CLR/
    HOME]"LI$;:PRINT
    "[CTRL+8][6
    ESPACIOS]HASTARDADO:
    [CTRL+5]"WW"SEG."
32Ø PRINT"[CTRL+8]ERRORES
    COMETIDOS:[CTRL+5]"S
33Ø IFK>3THENPRINT"[CRSR
    ABAJO]"TAB(7)INT(1Ø*NU/
    WW)"[CTRL+2]PALABRAS
    PORMINUTO."
34Ø PRINTLI$:GOTO53Ø
35Ø NU=Ø:P=Ø:S=Ø:FORWW=1
    TOMM:W$(WW)=WO$(INT
    (RND(1)*28)+1)
36Ø IFK=5ANDWW<>5
    THENW$(WW)=W$(WW)+""
37Ø NU=NU+LEN(W$(WW)):
    NEXTWW:TI$="ØØØØØØ"
    :FORWW=1TOMM:GOTO6Ø
38Ø POKE5328Ø;6:POKE53281,
    Ø:POKE198,Ø
39Ø PRINT"[CLR/HOME][
    1Ø*CSRSABAJO][CTRL+2]"
    TAB(15)"OPCIONES[CRSR
    ABAJO][CTRL+6]"
4ØØ FORZ=1TO5:PRINT
    TAB(13)Z;":TEST";Z:NEXTZ
41Ø PRINT
    TAB(12)"[CTRL+2][CRSR
    ABAJO]ENTRAROPCION..."
42Ø GETK$:K=VAL(K$):IFK<1
    ORK>5THEN42Ø
43Ø PRINT"[SHIFT+CLR/
    HOME]"LI$"[CTRL+6]"
44Ø IFK<3THENPRINT"[CRSR
    ARRIBA]PULSALATECLA
    INDICADACONEL
    ASTERISCO";
45Ø IFK=3THENPRINT"[CRSR
    ARRIBA]TECLEALALETRA
    QUEAPARECEEN
    PANTALLA.";
46Ø IFK=4THENPRINT"[CRSR
    ARRIBA]TECLEALAPALABRA
    QUESALEENPANTALLA[2
    ESPACIOS]";:MM=2Ø
47Ø IFK=5THENPRINTTAB
    (12)"[CRSRARRIBA]TECLEA
    ESTASPALABRAS":MM=5
48Ø PRINTLI$:FORZ=1TO
    1ØØØ:NEXT
49Ø PRINT
    TAB(11)"[CTRL+2]PULSA
    UNATECLAPARAEMPEZAR"
    :POKE198;Ø:WAIT198;1
5ØØ PRINTTAB(11)"[CRSR
    ARRIBA][29ESPACIOS]"
51Ø ONKGOTO5Ø,5Ø,5Ø,35Ø,
    35Ø
52Ø GOTO38Ø
53Ø FORZ=1TO1ØØØ:NEXT
    Z:GOTO38Ø
54Ø DATAR65Ø2,COM64,VIC2Ø,
    COM128,1ØSEG,MAR87,
    DOS5,1,27ABRIL,12DIAS,
    D8M3A87
55Ø DATA312KG,1836M,
    ORGCØØØ,SYS49152,
    LIST8Ø,GEN12,LINK32,
    MIL37,RS232C
56Ø DATADIN49152,CD4Ø27,
```



SN74123N,MC1488,
TIL111,M14412,XR32Ø,
ZN427,ICL8Ø38

Al ejecutar el programa, se te pedirá que selecciones uno de los cinco niveles ya familiares. Éstos son similares a los de antes, con la adición de los caracteres nuevos. En los niveles inferiores, se te pedirá que mezcles correctamente números, y algunas veces signos de puntuación, con las teclas existentes. Esto te resultará difícil, garantizando que no puedas concentrarte simplemente en los números.

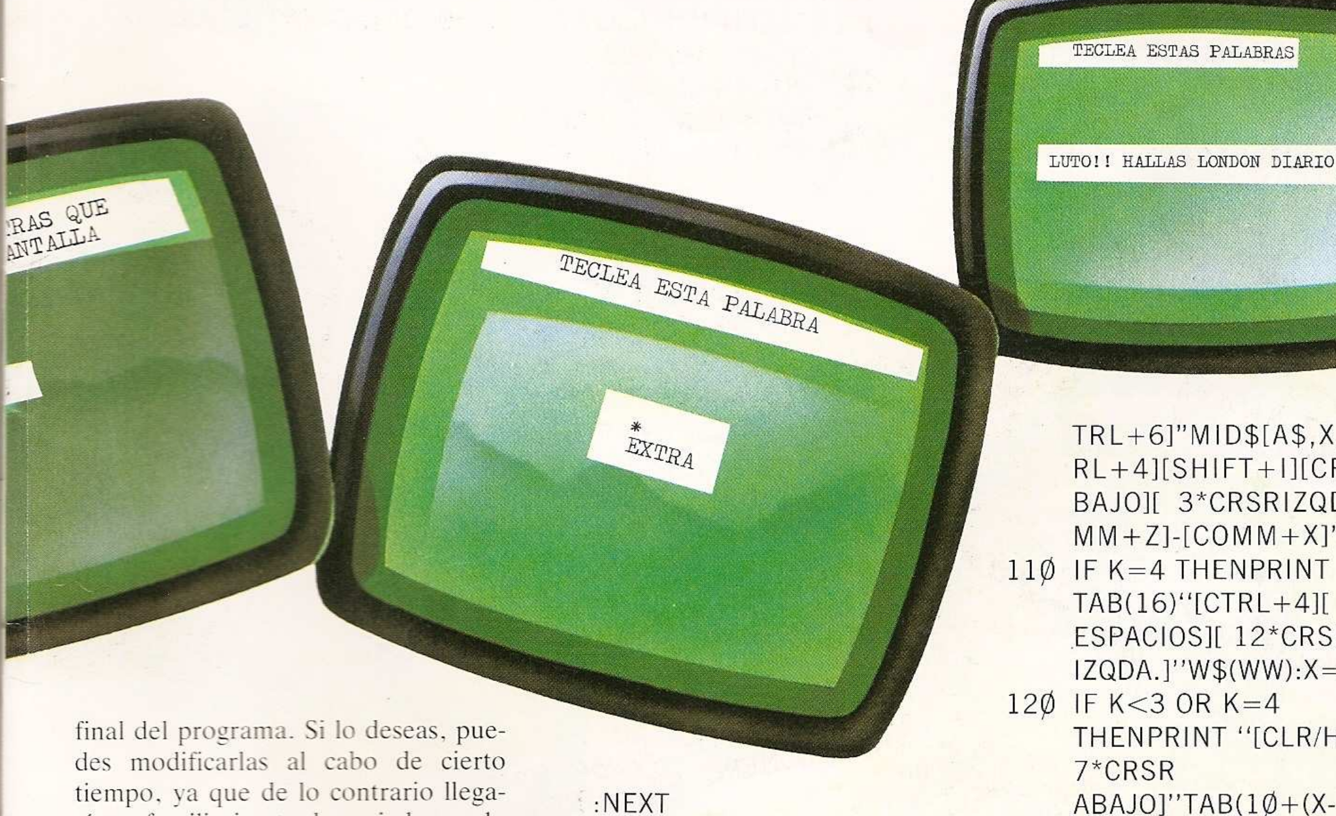
A continuación, en los niveles superiores, se te pedirá que teclees una mezcla de palabras, grupos de números y grupos formados por una mezcla de letras y números. Verás las palabras y los números seleccionados en las instrucciones DATA próximas al final del programa. Si lo deseas, puedes modificarlas al cabo de cierto tiempo, ya que de lo contrario llegarías a familiarizarte demasiado con lo que el ordenador va a pedirte que hagas y no se trataría ya de un auténtico desafío. Pero recuerda mantener el mismo número de palabras (o de grupos de caracteres) o el ordenador no podrá leer la cantidad correcta de datos.

Cuando domines las teclas de números, habrá llegado el momento de pasar a la siguiente lección. Esta te proporcionará la práctica necesaria para obtener los caracteres para los cuales se debe pulsar la tecla SHIFT.

LA TECLA SHIFT

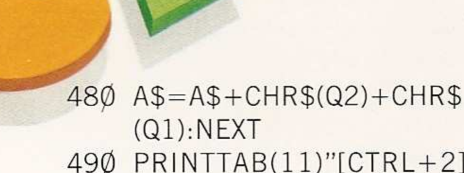
Esta vez, las líneas adicionales del programa te darán la posibilidad de utilizar la tecla SHIFT. Como antes, dichas líneas sustituirán a las líneas existentes o se incluirán entre ellas:

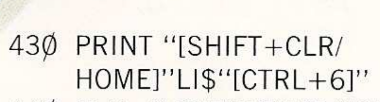
```
1Ø DIM W$(21); WO$(28):FOR
   Z=1 TO 28:READ
   WO$(Z):NEXT
2Ø FOR Z=1 TO4Ø:LI$=LI$+"-"
   :NEXT
   :LI$="[CTRL+2]"+LI$
3Ø PRINT "[SHIFT+CLR/
   HOME][CTRL+8]"TAB(7)
"***** MECANOGRAFIA *****"
4Ø PRINT
   CHR$(8)CHR$(14):POKE
   54296; 15:GOTO 38Ø
5Ø TI$="ØØØØØØ":S=Ø:WW=1
6Ø N=Ø:POKE 198,Ø
7Ø IF K=5 AND P>Ø
   THENN=N+1:P=P+1:GOTO
   14Ø
8Ø PRINT"[CLR/HOME][8*CRSR
   ABAJO]":IFK<3THENPRINT
   TAB(1Ø)"[CTRL+4][CTRL
   +9]"A$:POKE198;Ø
9Ø X=INT (RND(1)*2Ø)+1
   :N=N+1 :P=P+1:IF K=1
   THENX=N:GOTO 12Ø
1ØØ IF K=3 THENX=INT(RND
    (1)*20)+1:PRINTTAB(18
    )"[CTRL+4][COMM+A]-[CO
    MM+S][CRSR ABAJO][ 3*
    CRSR IZQDA.][SHIFT+I][C
    TRL+6]"MID$[A$,X,1)"[CT
    RL+4][SHIFT+I][CRSR A
    BAJO][ 3*CRSRIZQDA.][CO
    MM+Z]-[COMM+X]"
11Ø IF K=4 THENPRINT
    TAB(16)"[CTRL+4][ 12
    ESPACIOS][ 12*CRSR
    IZQDA.]"W$(WW):X=N+6
12Ø IF K<3 OR K=4
    THENPRINT "[CLR/HOME][
    7*CRSR
    ABAJO]"TAB(1Ø+(X-
    1))"[CTRL+2]*"
13Ø IF K=5 THENFOR Z=1 TO
    MM:PRINT
    "[CTRL+8]"W$(Z); :NEXT
    Z:PRINT :PRINT
    "[CTRL+9][CTRL+4]";
14Ø GET K$:IF K$="" THEN14Ø
```



```
150 IF K>3 AND
    K$=MID$(W$(WW),N,1)
    THEN210
160 IF K>3 THEN180
170 IF K$=MID$(A$,X,1)
    THEN210
180 W=54276:A=54277
190 POKE W, 33:POKE A,
    50:POKE 54273,
    30:S=S+1
200 POKE 54273, 0:POKE W,
    32:GOTO 140
210 W=54276:A=54277
220 IF K<3 OR K=4
    THENPRINT "[CLR/HOME][
    7*CRSR
    ABAJO]"TAB(10+(X-1))" "
230 IF K=5 THENPRINT K$;
240 POKE W, 33:POKE A,
    50:POKE 54273, 150
250 POKE 54273, 0:POKE W,
    32
260 IF K=1 AND N<20 THEN70
270 IF K> 3AND
    N<LEN(W$(WW)) THEN70
280 IF (K<4 AND K>1) AND
    N<20 THEN70
290 IF K>3 THENNEXT WW
300 WW=VAL(MID$(TI$,3,
    2))*60+VAL(RIGHT$(TI$,
    2))
310 PRINT
    "[CTRL+0][SHIFT+CLR/
    HOME]"LI$;:PRINT
    "[CTRL+8][6ESPACIOS]
    HASTARDADO:[CTRL+5]
    "WW"SEG."
320 PRINT "[CTRL+8]ERRORES
    COMETIDOS:[CTRL+5]"S
330 IF K>3 THENPRINT "[CRSR
    ABAJO]"TAB(7)INT (10*NU/
    WW)"[CTRL+2]PALABRAS
    POR MINUTO. "
340 PRINT LI$:GOTO 530
350 NU=0:P=0:S=0:FOR
    WW=1 TO
    MM:W$(WW)=WO$(INT
    (RND(1)*28)+1)
360 IF K=5 AND WW<>4
    THENW$(WW)=W$(WW)+" "
370 NU=NU+LEN(W$(WW))
    :NEXT WW:TI$="000000":
```

```
    FOR WW=1 TO MM:GOTO 6Ø
38Ø POKE 5328Ø, 6:POKE
    53281, Ø:POKE 198, Ø
39Ø PRINT "[CLR/HOME][
    1Ø*CRSR
    ABAJO][CTRL+2]"TAB(15)
    "OPCIONES[CRSR
    ABAJO][CTRL+6]"
4ØØ FOR Z=1 TO 5:PRINT
    TAB(13)Z;":TEST"; Z:NEXT Z
41Ø PRINT
    TAB(12)"[CTRL+2][CRSR
    ABAJO]ENTRAR OPCION..."
42Ø GET K$:K=VAL(K$):IF K<1
    OR K>5 THEN42Ø
```





```
43Ø PRINT "[SHIFT+CLR/
    HOME]"LI$"[CTRL+6]"
44Ø IF K<3 THENPRINT "[CRSR
    ARRIBA]PULSA LA TECLA
    INDICADA CON EL
    ASTERISCO";
45Ø IF K=3 THENPRINT "[CRSR
    ARRIBA]TECLEA LA LETRA
    QUE APARECE EN
    PANTALLA.";
46Ø IF K=4 THENPRINT "[CRSR
    ARRIBA]TECLEA LA
    PALABRA QUE SALE EN
    PANTALLA[ 2 ESPACIOS]";
    :MM=2Ø
47Ø IF K=5 THENPRINT
    TAB(12)"[CRSR
    ARRIBA]TECLEA ESTAS
    PALABRAS":MM=4
472 PRINT LI$:A$="":FOR Z=1
    TO 1Ø:Q1=65+RND(1)*26
473 Q2=35+INT (RND(1)*23)
474 IF RND(1)>.5Ø
    THENQ1=Q1+128
48Ø A$=A$+CHR$(Q2)+CHR$
    (Q1):NEXT
49Ø PRINTTAB(11)"[CTRL+2]
    PULSAUNA TECLA PARA
    EMPEZAR":POKE 198;Ø:
    WAIT198;1
5ØØ PRINTTAB(11)"[CRSR
    ARRIBA][29ESPACIOS]"
51Ø ONKGOTO5Ø,5Ø,5Ø,35Ø,
    35Ø
52Ø GOTO38Ø
53Ø FORZ=1TO1ØØØ:NEXT
    Z:GOTO38Ø
54Ø DATAR2D2,VIC2Ø,COM64,
    1A2B,15,2,87,EDAD16,
    HASTA7,TIENES2,A37NO,
    1SEG.
55Ø DATA9H53B,16/92,P5V,
    19A52,SALTA.5,1987,
    ZN432,7/12/1990,WE4AN1
56Ø DATAX24ZN,Q3C457,
    P9V3481,C6M4A,R3DY7L,
    651ØCPU,416RAM,QØ23AS,
    1HDPQØ
```

Cambia a modo minúsculas/gráficos para entrar las instrucciones DATA.

Ahora, los niveles inferiores del test te presentan todos los caracteres que están disponibles solamente cuando las teclas se pulsan junto con la tecla SHIFT, símbolos de puntuación, matemáticos, etc. Para hacértelo más difícil van mezclados con las letras originales de cada tecla, lo que significa que tienes que hacer ir y venir los dedos cada vez.

En los niveles superiores, tienes ahora una lista de palabras y de grupos de palabras al igual que antes, excepto que esta vez encontrarás mayúsculas y símbolos mezclados con minúsculas. Tu ordenador comprobará tu velocidad y tu precisión en dichas palabras y grupos de caracteres. Si ves que lo haces demasiado bien en estos ejemplos concretos, puedes en cualquier momento montarte tú mismo un nuevo conjunto de datos. Sin embargo, debes recordar que hay que conservar igual el número total de datos. Cuando puedas encontrar todos los caracteres del teclado, sin mirar y sin vacilar, podrás pasar a la sección siguiente.

JUEGO DE VELOCIDAD

Es el momento de pensar en incrementar tu precisión y tu velocidad. Uno de los mejores modos de lograrlo

consiste en teclear siguiendo los tiempos de un metrónomo, o algo similar, lo que mejorará tu regularidad y tu ritmo. A continuación, a medida que aumente tu destreza, puedes aumentar la rapidez del metrónomo y, por tanto, la de tus pulsaciones.

RECURRIENDO AL RELOJ INCORPORADO

Pero ¿por qué utilizar un metrónomo si tu ordenador dispone de un reloj incorporado? El siguiente programa es un completo y nuevo ejercicio de mecanografía dispuesto en forma de juego, en el cual tu puntuación dependerá de cuán bueno seas con el teclado. Consta de dos partes. La primera parte visualiza una línea de caracteres seleccionados de modo aleatorio, que tú deberás teclear en el mismo orden en que aparecen. La segunda parte es más difícil, ya que ahora los caracteres aparecen aleatoriamente en la pantalla uno a uno, de modo que no tengas ningún indicio de cuál es el que viene a continuación.

Antes de empezar el test, puedes seleccionar tu propio nivel de dificultad. Esto se lleva a cabo diciéndole al ordenador a qué velocidad quieres que se visualicen las letras, es decir cuántos caracteres por minuto deseas teclear. Entonces, el ordenador te dará un tiempo limitado en el cual debes teclear cada carácter, dándote en caso contrario un punto de error. En el primer nivel esto se traduce en un indicador móvil que te muestra qué letra deberías estar tecleando, mientras que, en el segundo nivel, es el propio carácter el que se muestra parpadeando solamente durante un intervalo de tiempo determinado.

Antes de que empieces el primer test, puedes elegir si deseas el teclado normal (sólo letras) o el teclado expandido (todos los símbolos). Además, antes de que empieces el segundo nivel (el test de caracteres), puedes decidir su duración. Se te pedirá cuántos caracteres deseas en total en el test. Cuando el ordenador los haya visualizado todos, se parará y te dará una puntuación basada en tus errores.

No es sólo la velocidad media lo que mide el ordenador en este test, ya que debes mantener un ritmo uniforme. Esto será realmente beneficioso para quien desee incrementar su velocidad general de escritura. Para ayudarte a adquirir la costumbre, el ordenador te dará una señal acústica junto al indicador visual correspondiente a cada letra.

Ahora teclea el programa y pon a prueba tu habilidad:

```
1Ø FOR Z=1TO 4Ø:LI$=LI$+"-
   ":NEXT :LI$="[CTRL+2]"
   +LI$
2Ø PRINT"[SHIFT+CLR/
   HOME][CTRL+8]"TAB(7)"***
   MECANOGRAFIA ***"
3Ø PRINT
   CHR$(8)CHR$(14):POKE
   54296, 15:GOTO 26Ø
4Ø S=Ø:WW=1:ER=Ø:NM=Ø:
   N=Ø
5Ø POKE5328Ø;5
6Ø PRINT"[CLR/HOME][5*CRSR
   ABAJO]";:IFK=1THENPRINT
   "[CTRL+4][CTRL
   +9]"TAB(1Ø)A$:POKE198,Ø
7Ø X=INT(RND(1)*20)+1:N=
   N+1:P=P+1:IFK=1
   THENX=N:GOTO9Ø
8Ø IFK=2THENX=INT
   (RND(1)*20)+1:PRINT
   TAB(18)"[CTRL+4][COMM+
   A]-[COMM+S][CRSRABAJO][
   3*CRSRIZQDA.][SHIFT+I]
   [CTRL+6]"MID$(A$,X,
   1)"[CTRL+4][SHIFT+I][CRSR
   ABAJO][3*CRSR
   IZQDA.][COMM+Z]-
   [COMM+X]"
9Ø IFK=1THENPRINT"[CLR/
   HOME][4*CRSR
   ABAJO]"TAB(1Ø+(X-
   1))"[CTRL+2]*"
1ØØ TI$="ØØØØØØ"
11Ø GETK$:IFK$=""ANDTI<TM
    THEN11Ø
12Ø IFTI>TM
    THENER=ER+1:GOTO
    17Ø
13Ø IFK$<>MID$(A$,X,1)
    THEN11Ø
14Ø IFK=1THENFORZ=4TO1
    STEP-1:PRINTLEFT$("[CLR/
    HOME][8*CRSRABAJO]",
    Z+6)TAB(1Ø+X-1)"
    [CTRL+2][CTRL+6][CRSR
    ABAJO][CRSRIZQDA.]
    ":NEXT
15Ø IFK=2THENPRINT"[CLR/
    HOME][5*CRSR
    ABAJO]"TAB(18)"[CTRL+4]
    [COMM+A]-[COMM+S]
    [CRSRABAJO][3*CRSR
    IZQDA.][SHIFT+I][CTRL+6]
    [CRSRDCHA.][CTRL+4]
    [SHIFT+I][CRSRABAJO][
    3*CRSRIZQDA.][COMM+Z]-
    [COMM+X]"
16Ø IFTI<TMTHEN16Ø
17Ø W=54276:A=54277:NM=
    NM+1:POKE53280,6
18Ø IFK=1THENPRINT"[CLR/
    HOME][4*CRSR
    ABAJO]"TAB(1Ø+(X-1))" "
19Ø POKEW,33:POKEA,
    5Ø:POKE54273,15Ø
2ØØ POKE54273,Ø:POKEW,
    32
21Ø IFK=1ANDN<2ØTHEN
    5Ø
22Ø IFK=2ANDN<TLTHEN
    5Ø
23Ø PRINT"[SHIFT+CLR/
    HOME][CTRL+Ø]"LI$"[CTRL
    +6]"KP"PULSACIONESPOR
    MIN.,HASFALLADO
24Ø PRINTER"VECESDEUN
    TOTALDE"NM
25Ø PRINTLI$
26Ø POKE5328Ø,6:POKE53281,
    Ø:POKE198,Ø
27Ø PRINT"[CLR/HOME][
    1Ø*CRSR
    ABAJO][CTRL+2]"TAB(15)"
    OPCIONES[CRSR
    ABAJO][CTRL+6]"
28Ø FORZ=1TO2:PRINT
    TAB(13)Z;":TEST";Z:NEXTZ
29Ø PRINT
    TAB(9)"[CTRL+2][CRSR
    ABAJO]QUEPRUEBADESEAS
    ?"
```

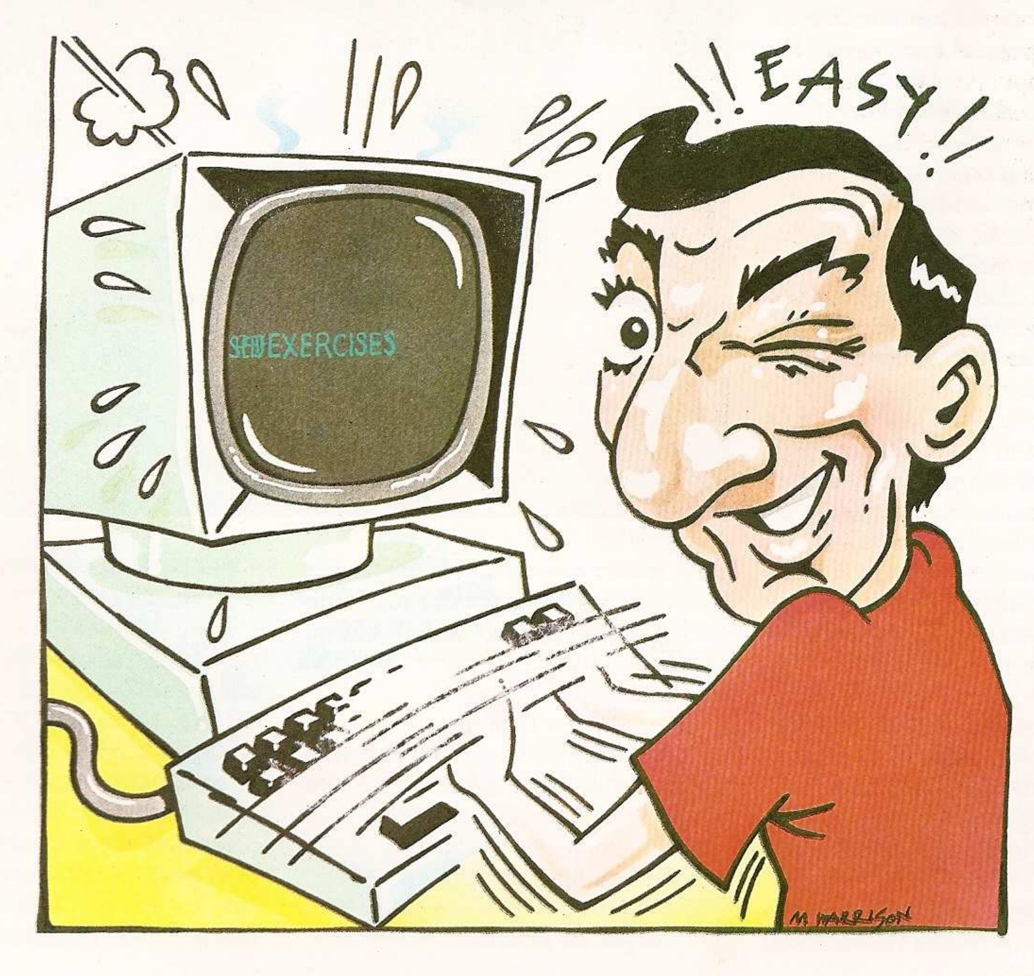
```
300 GETK$:K=VAL(K$):IF
    K<1ORK>2THEN300
310 INPUT"[SHIFT+CLR/
    HOME]N.DEPULSACIONES
    PORMIN.APRACTICAR";
    KP:IFKP<1THEN
    310
320 TM=3000/KP:INPUT
    "[SHIFT+CLR/
    HOME]LETRASNORMALESO
    AMPLIADAS(N/A)";
    NX$
330 IFNX$<>"N"AND
    NX$<>"A"THEN
    320
340 IFK=2THENTL=20:INPUT
    "[SHIFT+CLR/
    HOME]ENTRARNUMERODE
    CARACTERES";
    TL
350 PRINT"[SHIFT+CLR/
    HOME]"LI$"[CTRL+6]"
360 PRINT"[CRSR
    ARRIBA]PULSALATECLA
    CUANDOOIGASEL
    PITIDO"
370 SP=1:IFNX$="N"
    THENSP=.5
380 PRINTLI$:A$="":FOR
    Z=1TO10.5STEPSP:Q1=65
    +RND(1)*26
390 Q2=33+INT(RND(1)
    *25)
400 IFRND(1)>.50
    THENQ1=Q1+
    128
410 IFNX$="N"
    THENA$=A$+CHR$(Q1):
    NEXT:GOTO430
    NEXT:GOTO430
420 A$=A$+CHR$(Q2)+CHR$
    (Q1):NEXT
430 PRINTTAB(11)
    "[CTRL+2]PULSAUNA
    TECLAPARA
    EMPEZAR":POKE198,
    0:WAIT198,1
440 PRINTTAB(11)"[CRSR
    ARRIBA][28ESPACIOS]"
450 GOTO40
```

MICROCONSEJO ACERCA DE LA RAPIDEZ

Cuando utilices el programa de velocidad, es mejor empezar por el teclado normal a baja velocidad, digamos de 30 a 50 caracteres por minuto.

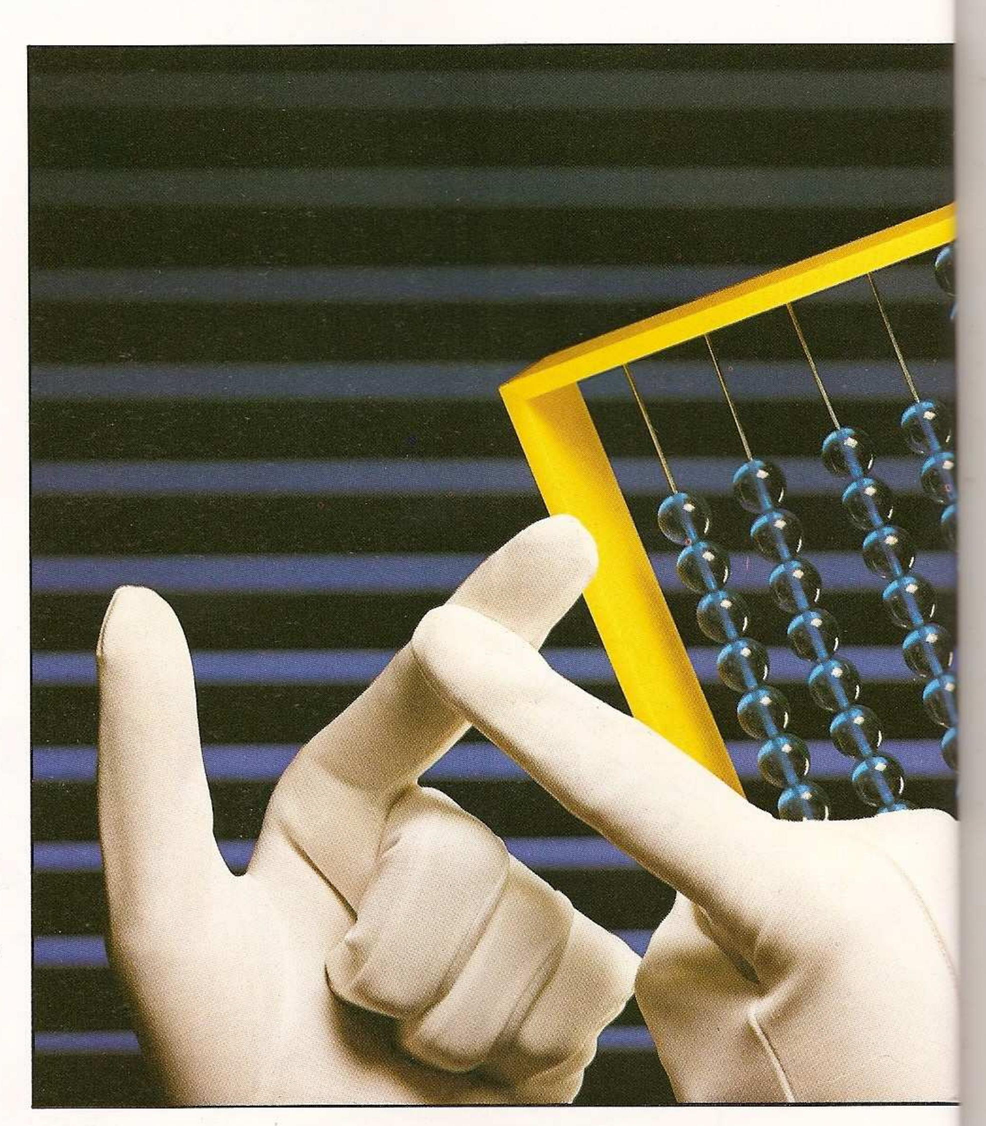
El riesgo a evitar consiste en teclear los caracteres familiares más rápidamente y luego ir más despacio cuando te encuentres con caracteres sobre los que no estés tan seguro, particularmente si eliges el teclado expandido.

Una vez tengas el hábito de teclear a un ritmo constante puedes seleccionar el teclado expandido y, a continuación, incrementar gradualmente tu velocidad.



EVITANDO ERRORES

Incluso cuando un programa está totalmente depurado, todavía pueden aparecer errores. Esto se debe al uso o al abuso que se hace del propio programa. Éste es el tema del que nos ocuparemos a continuación.



La palabra clave en este caso es *uso*. Independientemente de lo bien hecho que esté un programa, desde el punto de vista exclusivamente técnico, si de alguna manera su uso resulta difícil, confuso o engañoso, es seguro que en algún momento alguien tendrá problemas con el mismo. Y si alguien tiene problemas al utilizar un ordenador, lo más probable es que se produzca algún tipo de error.

En realidad el secreto consiste en hacer que el programa sea agradable de usar, de forma que cada una de las fases del mismo sea convenientemente anunciada, y el usuario aborde cada nueva acción con pleno conocimiento de lo que está haciendo.

Esto quiere decir que hay que disponer una gran cantidad de representaciones y mensajes de pantalla, y que hay que proteger tanto al usuario como al programa de la posibilidad de cometer errores.

Siempre que sea posible, merece la pena incluir en el programa una protección contra todo tipo de entrada accidental. Las pulsaciones erróneas de tecla, el pulsar dos teclas a la vez, las entradas ilegales, los valores imposibles, todo ello significa la muerte de ciertos programas a menos que se adopten precauciones especiales. Para hacer todo esto, tienes que incorporar en tus programas varios tipos de rutinas de comprobación de errores y validación de entradas. De hecho, muchos de los programas que hemos visto ya en **INPUT** utilizan rutinas de esta clase.

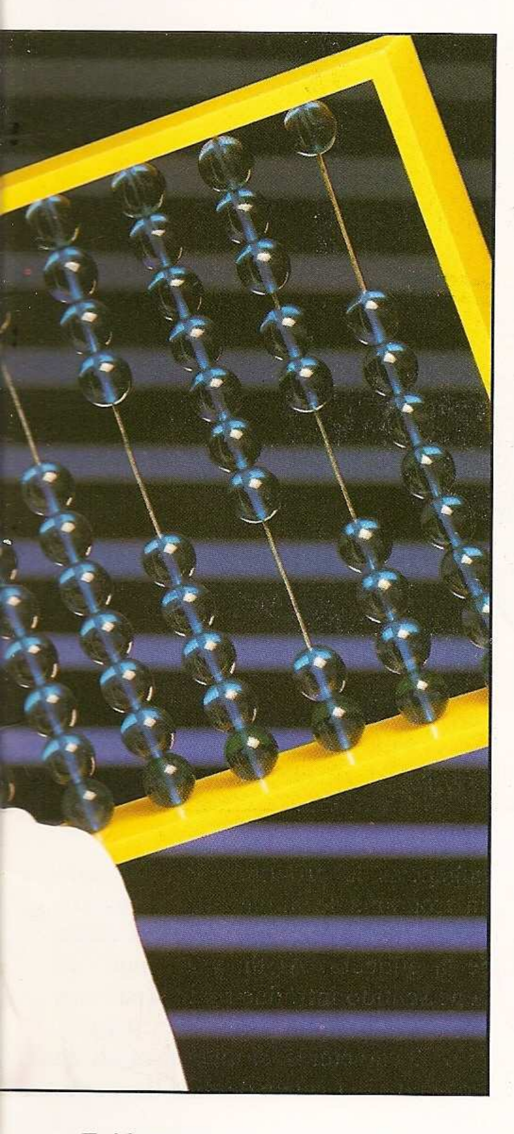
El uso de unos mensajes de pantalla que sean precisos resulta fundamental para ayudar al usuario a entender perfectamente cómo puede responder cuando se le está invitando a que seleccione una opción, por ejemplo en un menú.

Imagínate por ejemplo que tenemos un menú con las siguientes opciones, típicas de la parte inicial de un programa sencillo de base de datos:

1 Crear un nuevo registro
2 Modificar un registro ya existente
3 Cargar un fichero
4 Almacenar un fichero
... etcétera.

Si a continuación se pone un mensaje que diga algo así como "SELECCIONE UNA OPCION" o "HAGA SU ELECCION", el usuario se queda en realidad un poco en tinieblas respecto a la forma en que tiene que hacer lo que venga a continuación. ¿Deberá pulsar el número de opción, o deberá teclear más bien una o varias letras del correspondiente mensaje?

- USO DE MENSAJES ADECUADOS
- A LA CAZA DE ERRORES
- EXCLUSIÓN DE LOS VALORES NO VÁLIDOS
- INDICACIÓN DE ERRORES



Evidentemente, un mensaje mucho más satisfactorio en este sentido sería "SELECCIONE OPCION (1-4)" o bien "PARA INDICAR SU ELECCION PULSE UNA TECLA DEL 1 AL 4".

De la misma forma, en los mensajes que requieran una respuesta sencilla que sea un *sí* o un *no*, indícalo en el propio mensaje. Por eso, en lugar de utilizar un mensaje que, por ejemplo, diga "¿QUIERES JUGAR OTRA VEZ?" y en el que el *sí* podría estar indicado por la pulsación de una tecla cualquiera no específica, o por la pulsación del botón de *fuego* del *joystick*, o por la tecla Y, es mucho mejor que especifiques las opciones. Así, podrías poner "PARA JUGAR OTRA VEZ, APRIETA EL DISPARADOR" o bien "¿QUIERES JUGAR OTRA VEZ (S/N)?". Cualquiera de las dos posibilidades es mucho más directa y da una idea mucho más clara de la manera de continuar.

Incluso el mensaje "PULSAR CUALQUIER TECLA PARA CONTINUAR" resulta mejor a este respecto. Pero este mensaje tan conocido y tan popular también puede ser mejorado especificando una tecla determinada; con ello se evita que al usuario se le ocurra pulsar precisamente **STOP**, **BREAK**... y pueda salirse del programa. Seguro que no hay confusiones si utilizas un mensaje del tipo: "PARA CONTINUAR, PULSA C".

En algunos ordenadores es posible destacar las pulsaciones requeridas utilizando el vídeo inverso, lo cual constituye una alternativa bastante atractiva.

Con independencia de los tipos de mensajes utilizados y de las pulsaciones de tecla disponibles, constituye una buena práctica de programación el inutilizar todas las teclas que puedan originar la parada del programa, y el utilizar rutinas que impidan todas las entradas imposibles, como se explica más adelante. Resumiendo, en las opciones a las que hay que responder con un simple *sí* o *no*, podrías utilizar algo como lo siguiente:

```
90 GET A$
95 IF A$<> "S" AND A$<>"N"
   THEN 90
```

Otra forma de aligerar el trabajo del usuario consiste en restringir la cantidad real de información que hay que introducir en un punto determinado. Si con la pulsación de una sola tecla se puede hacer una determinada tarea, úsala. Y para evitar confusiones, conviene que a lo largo de todo un programa controlado a base de menús utilices de forma sistemática el mismo tipo de mensajes y respuestas. De esta forma, si la selección del menú de apertura se hace tecleando un solo número, procura utilizar este mismo sistema para los otros menús que vengan detrás.

Insistiendo aún más en este punto, utiliza las mismas convenciones para cada menú y para cada lista de opciones. Si por ejemplo en uno de los menús la tercera opción es *GUARDAR* mientras que en otro menú la tercera opción es *SALIR*, lo más probable es que más de uno se sienta incómodo con tu programa en el futuro...

En los casos en que se requieran entradas de datos se puede aplicar la misma regla, especialmente cuando intervienen secuencias más bien largas. Los mensajes de entrada múltiple quedan bien cuando los datos están restringidos a un conjunto de campos muy sencillos, como ocurre por ejemplo en el caso de un fichero de nombres y direcciones, en el que invariablemente cada nombre va seguido por cuatro líneas de dirección y un número de teléfono.

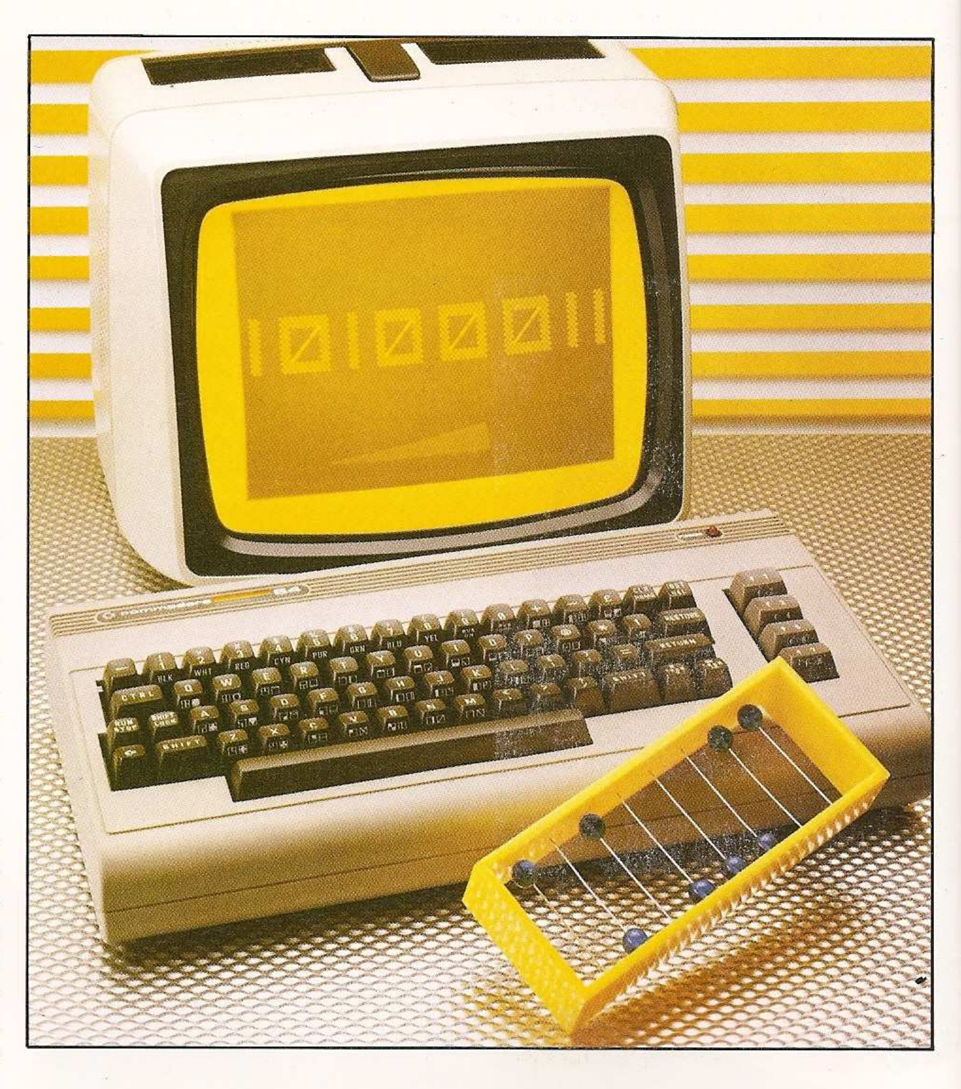
Pero el caso de un fichero con registros de clientes es ya algo más complejo y requiere bastante más cuidado. En este caso es bastante probable que el número de entradas difiera de un registro a otro y que algunos campos tengan que dejarse en blanco.

Puedes reducir los errores clasificando las entradas requeridas en grupos lógicos. Así puedes seguir conservando un único mensaje para el nombre y la dirección, dedicando a continuación mensajes especiales para detalles específicos.

Organiza las cosas de modo que los mensajes aparezcan ellos solos en la pantalla, o en un color diferente, o por lo menos bien separados del mensaje anterior. El borrado de toda la pantalla después de cada grupo de mensajes es algo que el ojo aprecia y agradece mucho más que un simple *scrolling* hacia arriba.

Llegados a este punto tenemos que hacer otra observación: en algunos programas, una entrada inesperada puede llamar a su propia subrutina. Por ejemplo, en un fichero de datos, se pueden requerir elementos de información adicionales para determinadas entradas. Si ahora resulta que el usuario, debido a la razonable familiaridad que ya ha ido adquiriendo con la secuencia de mensajes de pantalla, no presta la debida atención a la nueva «ramificación» de los mensajes y se cree que nada ha cambiado, puede encontrarse con toda clase de errores.

En casos como éste son esenciales las rutinas para la caza de errores, pero conviene de todas formas prevenir de alguna forma al usuario de que las condiciones de entrada han cambiado. Una simple intermitencia en la pantalla, o la presentación del mensaje en vídeo inverso son normalmente más que suficientes. También podrías incorporar algún tipo de alarma audible, tal como un *beep*.



LA CAZA DE ERRORES

La forma más sencilla de evitar que los errores sean *absorbidos* por un programa es brindar al usuario la opción final de que acepte o rechace lo que ya ha sido introducido. Puedes hacer esto utilizando una rutina que envíe un mensaje más o menos como éste: "¿ESTAS SEGURO? (S/N)". Al pulsar *N* simplemente vuelve a empezar la rutina de introducción de datos, mientras que con *S* pasa a actuar sobre la información ya introducida. Pero esto sólo es realmente necesario en las rutinas de entradas muy largas.

Si decides incorporar dentro de un programa una rutina de aceptación de entradas, intenta combinar todas las respuestas en un solo grupo. Así resulta de lectura mucho más fácil y es mucho mejor que andar repitiendo toda la secuencia de preguntas y respuestas individualmente.

Aunque las rutinas de aceptación de entradas proporcionan un método sencillo de evitar errores, los programas tienen que estar protegidos contra las entradas incorrectas.

Por ejemplo: ¿pueden introducirse letras y números en algún punto de un programa en el que sólo se permiten letras, o sólo números? Tienes que procurar anticiparte siempre a problemas de este tipo cuando construyas las rutinas de entrada de tus propios programas.

LÍMITES DE LONGITUD

En la mayoría de los programas de ficheros de datos, las longitudes de las entradas tienen que adaptarse a las exigencias del programa. Por ejemplo un programa de etiquetas tiene sus entradas restringidas por el tamaño físico de la etiqueta. Al fin y al cabo, no tiene sentido introducir una línea con una dirección que tenga, por ejemplo, 25 o 30 caracteres, si una línea así de larga no va a caber dentro de la etiqueta.

Puedes servirte en estos casos de un indicador que sugiera el límite real de las entradas, como por ejemplo utilizando el vídeo inverso o cualquier otro artificio visual que sugiera los límites efectivos de cada campo.

Incluso en estos casos se necesita una programación adicional que invalide las entradas demasiado largas. En algunos casos tal vez sea preferible cortar por las buenas las líneas que sean demasiado largas; este tipo de truncamiento por el sitio adecuado

está basado en la propia rutina de aceptación de entradas. Pero normalmente es mejor iniciar de nuevo la secuencia de entrada en el punto en que se produjo el error, enviando un mensaje de error que diga algo así como "ENTRADA DEMASIADO LARGA" seguido de un mensaje que diga: "ESCRIBALO OTRA VEZ", o cualquier otra cosa que se te ocurra.

Otra de las razones principales para fijar límites a las longitudes de las entradas, dentro de un programa como un fichero de datos, es la conservación de la memoria. Después de todo, de nada sirve disponer 28 caracteres para una entrada de un apartado de correos, cuando la máxima longitud requerida por la misma nunca supera los 8 caracteres. Ajusta las longitudes de tus campos todo lo posible para ahorrar memoria y obtener el mayor rendimiento de tus ficheros de datos.

VALORES NO VÁLIDOS

El poner límites es también importante por otras razones, particularmente en los programas que hacen uso de una entrada numérica para cálculos posteriores. Por ejemplo, supongamos que el ordenador te pide tres números (o busca esos tres números en la memoria) y divide la suma de los dos primeros por el tercero. Una leve imprecisión de uno de tus dedos puede hacer que el tercer número introducido sea un 0 en lugar de un 9. O también puede ocurrir que como resultado de un cálculo interno realizado por tu ordenador, el valor 0 sea asignado a la tercera variable.

A menos que tu programa esté adecuadamente protegido, el resultado será un mensaje de error de *división por cero* con una detención brusca del programa.

En un caso como éste, la protección que se necesita es mínima, en el supuesto de que sólo se trate de entradas desde el teclado; basta con poner una sencilla rutina de comprobación de teclas para restringir el margen de valores aceptables.

Pero para los cálculos «internos» en los que podría resultar un valor cero como dato para otros cálculos posteriores, hay que utilizar métodos específicos de captura de errores. Se trata de un caso típico en que hay que anticiparse poniéndose en el caso peor, incorporando en el programa una rutina de comprobación de errores que utilice operadores relacionales. Podrías añadir algo más o menos como esto:

```
IF A=Ø THEN GOTO 1ØØØØ
```

La línea 10000 sería en este caso el principio de una rutina que podría modificar el valor de 0, asignándole otro valor que, aunque muy próximo a cero, sea aceptable por el programa; también podría ser que avisara al usuario de que está a punto de producirse un cálculo no válido. En el último ejemplo el programa podría ser reconducido a una rutina de entrada adecuada que permita introducir un valor alternativo.

Para comprobar que un dato se encuentra dentro de un determinado margen de valores, puedes utilizar algo tan simple como lo siguiente:

```
IF N>1ØØ OR N<1 THEN...
```

En el caso de que el valor introducido esté fuera de márgenes, la línea anterior enviaría al programa al principio de la rutina de entrada. En el caso de que exista el riesgo de que se produzca una entrada no válida bajo control directo del usuario del programa, te conviene utilizar mensajes de pantalla adecuados que informen del margen de valores disponibles y además, en caso de que se produzcan errores, informe al usuario de dónde está el problema. Todos estos mensajes pueden estar contenidos en una subrutina a la que se accede cada vez que sea necesario, como veremos más adelante.



BORRADO DEL BUFFER

Hay algunos ordenadores, y el Commodore es uno de ellos, que tienen que utilizar una zona de almacenamiento temporal para las entradas y el teclado. Esto se hace por medio del llamado *buffer de entrada* o *buffer de teclado*. Puede ser que se produzcan errores si accidentalmente se introducen caracteres equivocados o todavía están almacenadas en el buffer las anteriores pulsaciones de tecla.

Por ejemplo en un programa que acepte una secuencia de caracteres individuales, al pulsar una tecla por segunda vez, se envía al buffer de teclado un segundo carácter, probablemente indeseado, quedando listo para su procesamiento. El primer mensaje de pantalla acepta el primer carácter, mientras que el segundo difícilmente va a tener tiempo de presentar de nuevo el carácter antes de que efectivamente quede asignado como valor de la segunda pulsación de tecla, con lo cual el programa proseguirá su marcha.

Para evitar este tipo de situaciones, es aconsejable incluir la siguiente sentencia como precaución adicional justamente antes de una secuencia de entrada:

```
POKE 198,Ø
```

Con esto, el buffer de teclado queda completamente borrado antes de que llegue la secuencia de entrada que se espera inmediatamente a continuación.

INDICACIÓN DE ERRORES

Si al usuario se le proporcionan instrucciones claras de lo que tiene que hacer cuando se enfrenta con la introducción de una secuencia de datos, ya se están eliminando de entrada una gran cantidad de errores posibles. No obstante siempre seguirán produciéndose errores, en cuyo caso una indicación clara del problema ayudará enormemente a que éste no vuelva a producirse.

Lo que se necesita en este caso no son los mensajes de error que te envía habitualmente tu ordenador, sino un conjunto de mensajes de error redactados *a medida*. Todos estos mensajes de error pueden ir en un programa que utilice una rutina especial a la que se llame cada vez que se produzca un problema.

Entre las aplicaciones típicas de esta rutina estarían los resultados numéricos de un cálculo o de una entrada exterior que se salgan de los límites permitidos, la duplicación de nombres en un campo clave, la longitud incorrecta de una entrada, o una entrada de tipo distinto del esperado, etc. De hecho, puedes imaginarte todos los tipos de problemas que pueden presentarse en un programa y preparar un mensaje de error para cada uno de ellos.

Utilizando una matriz y una organización adecuada de tus variables, puedes definir el número de mensajes de error requerido para permitir al programa asignar el valor de la variable en función de los errores que se produzcan.

La matriz correspondiente a estos mensajes de error se dimensionaría al principio del programa como parte del procedimiento de inicialización. Naturalmente, habría que efectuar de vez en cuando modificaciones periódicas al objeto de incluir los nuevos posibles errores que se vayan descubriendo a medida que se va desarrollando el programa. Así, por ejemplo, si estás pensando en utilizar un conjunto de nueve programas de mensajes de error, cerca del comienzo de tu programa puedes utilizar una sentencia DIM como se hace en este programa:

```
1Ø DIM EM$(9): FOR Z=1 TO
   9:READ EM$(Z):NEXT Z
2Ø DATA "¡ENTRADA
   DEMASIADO LARGA!",
   "¡PULSACION
   INCORRECTA!"
22 DATA "¡CONTRASEÑA
   INCORRECTA!", "¡SIN
   DATOS!"
24 DATA "¡REINTRODUZCA LOS
   DATOS!", "¡NO TOCAR!"
26 DATA "PULSE (S)I O (N)O",
   "¡SOLO NUMEROS!", "¡SOLO
   LETRAS!"
```

Es evidente que puedes definir tus propios mensajes de error de manera que se adapten lo mejor posible a las condiciones de tu programa. Después, a lo largo del programa, tendrás que ir intercalando las comprobaciones de los valores introducidos:

```
1ØØØ EM=Ø: INPUT A$
1Ø1Ø IF LEN (A$)>25 THEN
     EM=1
```

En el ejemplo anterior, la línea 1010 es opcional y puede ser remplazada por muchas alternativas, dependientes del programa de que se trate. Aquí tienes algunas de ellas:

```
1Ø1Ø IF A$<> "CREDITO"
     THEN EM=3
1Ø1Ø IF A$="5" OR A$="9"
     THEN EM=4
1Ø1Ø IF A$="" THEN EM=5
1Ø1Ø IF A$=" " THEN EM=6
1Ø1Ø IF A$<> "S" AND
     A$<>"N" THEN EM=7
1Ø1Ø IF A$<"Ø" OR A$>"9"
     THEN EM=8
1Ø1Ø IF A$<"A" OR A$>"Z"
     THEN EM=9
1Ø1Ø FOR Z=1 TO LEN (A$): IF
     MID$(A$,Z,1)="O" THEN
     EM=2
1Ø15 NEXT Z
```

Como puedes ver, todas las variantes propuestas llevan el mismo número de línea de programa, ya que en una rutina particular sólo debe figurar una de ellas.

Se exceptúa el caso del último ejemplo presentado, que por tratarse de un bucle requiere una sentencia NEXT en una línea posterior, línea 1015 en este caso.

Cuando al comprobar una pulsación de tecla se revela la presencia de un error, la variable EM adopta un valor particular que depende del mensaje de error específico, tal como se haya definido anteriormente como parte de la matriz EM.

La primera opción propuesta sirve para examinar si el dato introducido contiene más de 25 caracteres, la segunda permite comprobar la ortogra-

fía de la palabra clave. La tercera opción, que encontraría una aplicación típica en un menú de selección, señala que no hay datos disponibles si se eligen las opciones 5 y 9. La cuarta responde al usuario con cierto descaro en el caso de que éste pulse la barra espaciadora. La quinta opción es una variante de la comprobación de tecla que se suele poner habitualmente después de un mensaje de los del tipo Sí/No. Las dos opciones siguientes sirven para comprobar que sólo se introducen letras, o sólo números y la última es para cerciorarse de que no se ha colado en el programa una O en lugar de un 0.

El programa regresaría después al punto en el que se produjo la entrada de datos incorrecta.

A continuación el programa avanzaría, mediante una subrutina, hasta una rutina de presentación de mensajes que tendría la siguiente forma:

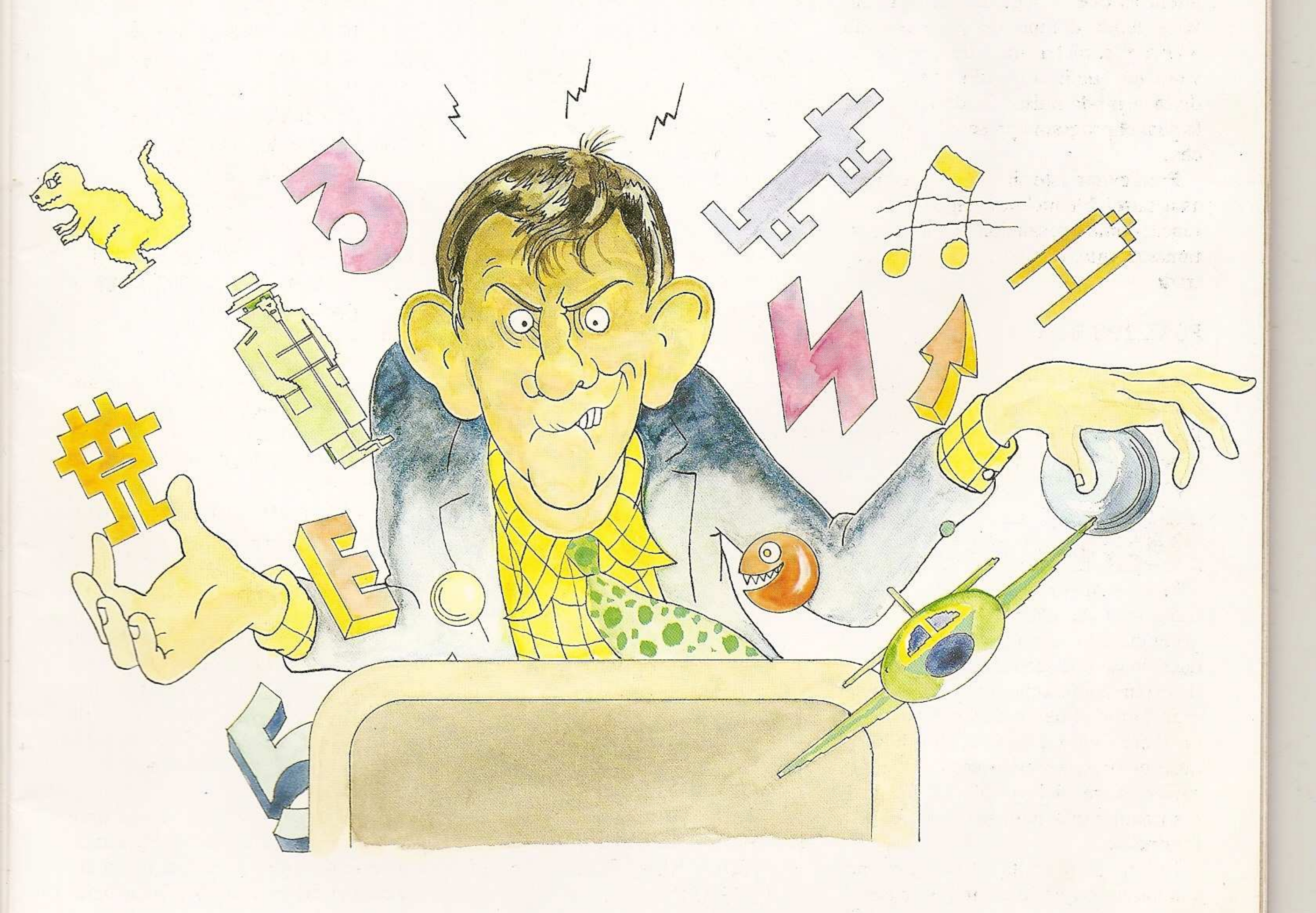
```
2ØØØ IF EM>Ø THEN PRINT
     EM$(EM)
```

A PRUEBA DE BOMBAS

Ya hemos visto antes la forma en que puede realizarse la comprobación de las pulsaciones de tecla para validar las entradas de un menú. Aunque el procedimiento descrito sirve para proteger el programa de las entradas incorrectas, hacen falta otros refinamientos para tener el programa bien protegido durante la ejecución.

Por ejemplo, imagínate un programa escrito para ser utilizado por niños pequeños. Lo ideal sería que sólo hiciera falta pulsar una tecla, o en todo caso dos o tres, para poder ir avanzando en el programa. Todas las demás teclas de letras y números pueden ser invalidadas de una forma bastante fácil. Pero ¿qué sucede con las teclas de control tales como *RUN/STOP*, *RESTORE*, *BREAK* o *ESCAPE*? Hay varios *POKEs* disponibles que puedes utilizar para inhibir algunas de las teclas más sensibles de los *Commodores*. Para el 64, puedes inhibir la tecla *RUN/STOP* con *POKE 808,239*. Para inhibir *RUN/STOP* y *RESTORE*, utiliza *POKE 808,251*. Para volver a habilitar cualquiera de las dos, teclea *POKE 808,237*.

En el caso del Vic 20, puedes inhibir *RUN/STOP* y *RESTORE* con *POKE 808,128*. La activación se hace con *POKE 808,112*.



ABISMO: PROGRAMA UN JUEGO COMPLETO

¿Puedes aprender el lenguaje ensamblador antes de que las cabras se coman tu merienda? Empieza ya a programar un magnífico juego en código máquina que iremos completando en sucesivos números de INPUT. Así combinaremos el aprendizaje de código máquina con la confección de un juego casi comercial.

Existen muchas aplicaciones comerciales serias para la programación en **lenguaje máquina**. Pero la programación de juegos te permite poner de manifiesto todos los principios importantes; además, el aprendizaje de la programación en el árido **código máquina** es mucho más divertido cuando se aplica para configurar un juego.

Por eso vamos a ver en INPUT un juego completo en **código máquina**, construido especialmente para aprovechar al máximo todas las posibilidades de programación del Commodore 64. El juego te mostrará cómo se va construyendo un juego típico y cómo se combinan los diferentes elementos de programación para producir gráficos interesantes, una acción suave y efectos emocionantes.

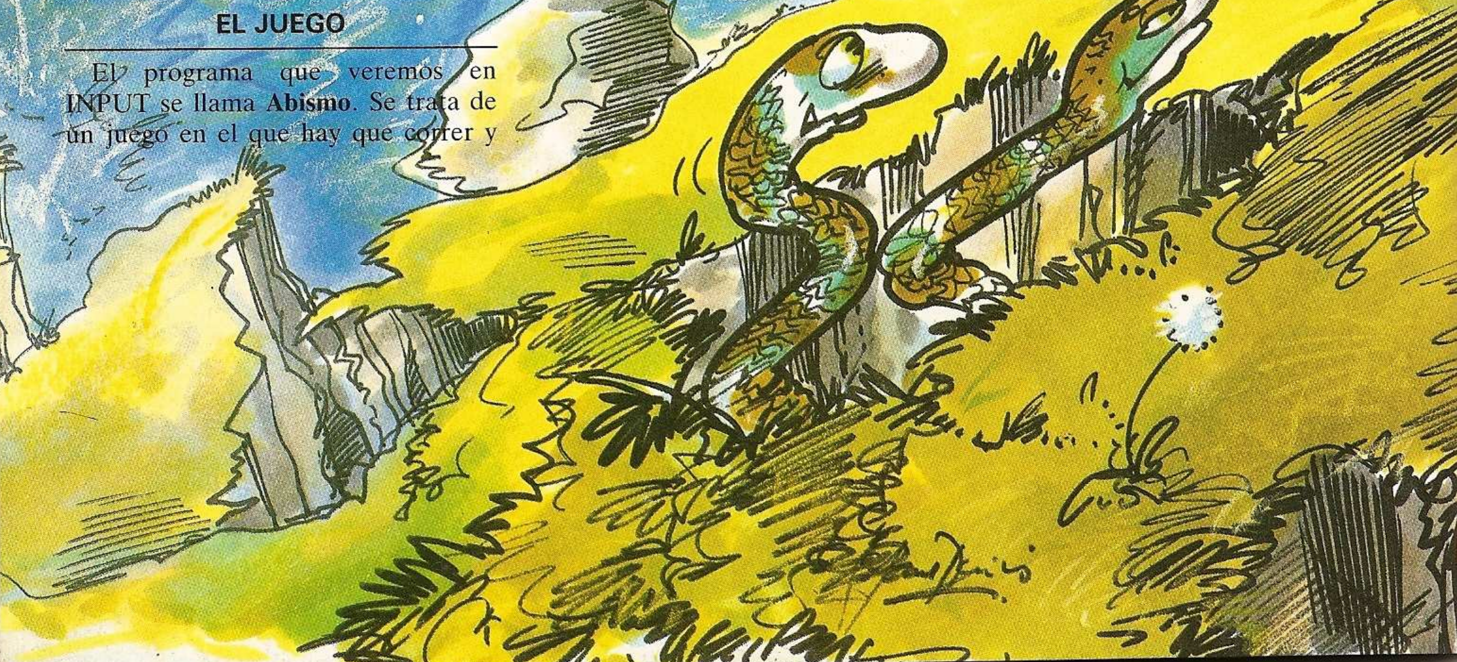
EL JUEGO

El programa que veremos en INPUT se llama **Abismo**. Se trata de un juego en el que hay que correr y saltar, en el que aparecen cuatro pantallas que van siendo progresivamente más difíciles. El personaje principal, que ha ido a merendar a un acantilado que hay junto al mar, ha decidido ir a dar un paseo para que se le abra el apetito, pero cuando regresa se encuentra con que unas cabras han esparcido toda su merienda por las rocas. Te ves obligado a escalar hasta lo alto del acantilado para recuperar tus posesiones perdidas. La cosa es todavía más urgente por el hecho de que la marea está subiendo y estás en peligro de ahogarte si no consigues alcanzar a tiempo la parte más alta del acantilado.

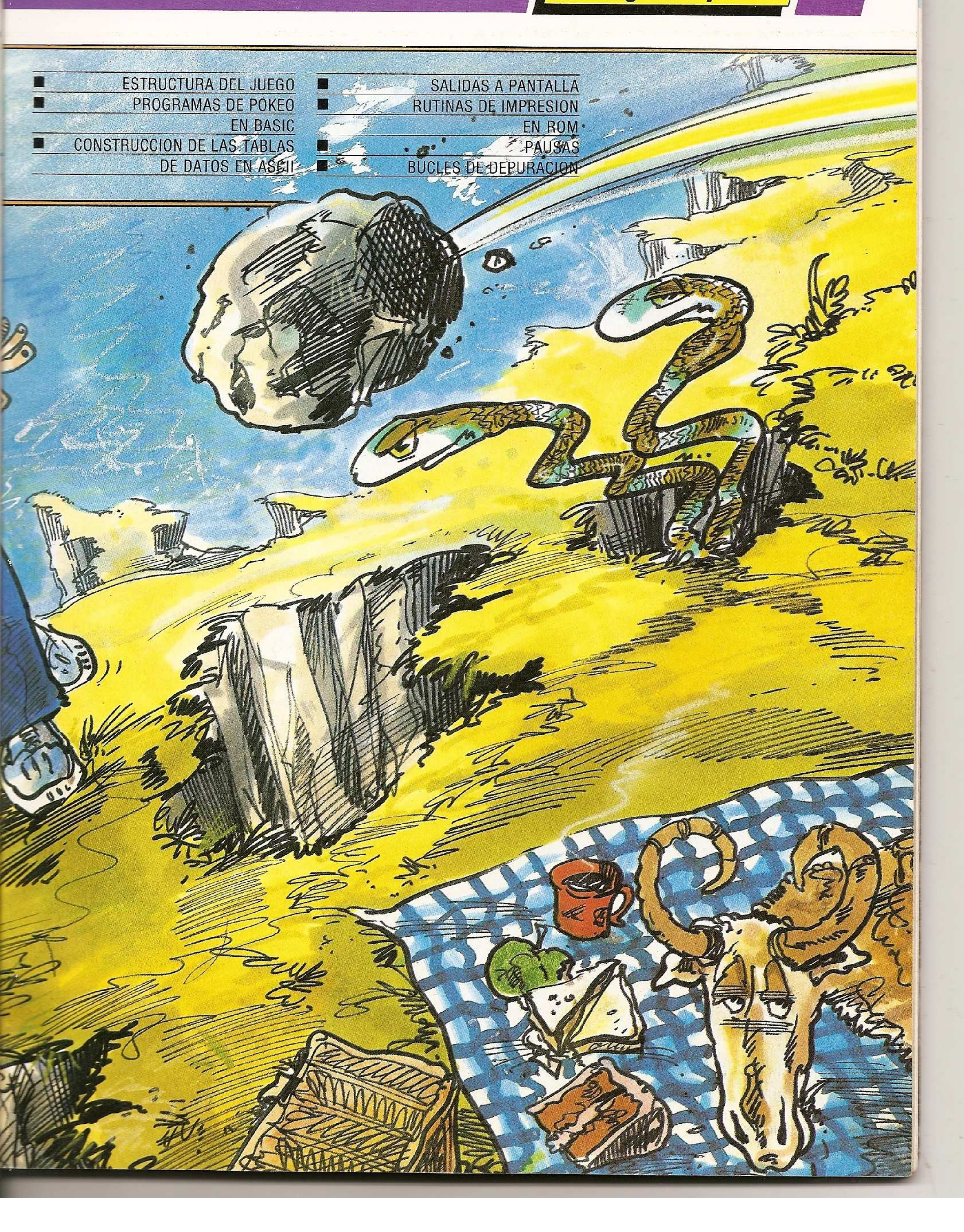
En la primera pantalla te ves estorbado por las rocas que caen. Las rocas van cayendo por la pendiente y te ves obligado a ir saltando sobre ellas. El menor resbalón puede significar tu muerte súbita. Puedes controlar tus saltos y tu carrera actuando sobre las teclas N y M. Si resultas alcanzado por una piedra, te verás inmediatamente sepultado. ¡Menos mal que tienes cinco vidas!

Cuando llegas hasta lo alto del acantilado y reclamas el primer artículo de tu perdida merienda, nuevamente desciendes hasta abajo y te encuentras en la segunda pantalla. Cuando intentas escalar esta vez el acantilado, te en-





- ESTRUCTURA DEL JUEGO
- PROGRAMAS DE POKEO EN BASIC
- CONSTRUCCION DE LAS TABLAS DE DATOS EN ASCII
- SALIDAS A PANTALLA
- RUTINAS DE IMPRESION EN ROM
- PAUSAS
- BUCLES DE DEPURACION



cuentras con que tienes que saltar por encima de unas grietas que te vas encontrando. Si caes en cualquiera de ellas, te verás nuevamente sepultado.

Cuando consigues llegar a lo alto de la segunda pantalla, pasas a la tercera. También ahora tienes que escalar la pendiente, evitando las grietas. Pero esta vez las grietas están habitadas por serpientes venenosas que harán todo lo posible por morderte cuando pases por encima. En la cuarta pantalla, tendrás que enfrentarte con grietas, serpientes y cantos rodados.

En todas las pantallas tienes, además, el problema de la marea, que sube sin cesar. Vas ganando puntos a medida que vas ascendiendo por la pendiente, además de una bonificación si consigues reunir, sin perder ni una vida, cuatro objetos que formen parte de tu merienda.

Aparte de su carácter formativo para el aprendizaje de la programación, *Abismo* resulta de lo más divertido de jugar. Lo iremos publicando por etapas en números sucesivos de INPUT. Explicaremos con todo detalle cómo funciona cada parte y cómo se adapta en la estructura global del juego. También veremos ejemplos de cómo se pueden utilizar algunas de las rutinas para diversas aplicaciones.

Al final dispondrás de un juego similar a muchos de los que existen en el mercado.

ESTRUCTURA DEL JUEGO

El fondo del escenario y los personajes móviles se construyen por medio de GDUs. Para generar el fondo se utilizan los correspondientes bucles que se encargan de irlo presentando en pantalla.

En la primera pantalla se superponen las grietas y las serpientes. De esta manera, no hace falta redefinir otra vez la mayor parte del fondo en las pantallas segunda y tercera.

La parte principal del programa incluye una rutina de ejecución que controla el tiempo y la prioridad de los sucesos que ocurren. Dichos sucesos se incorporan en forma de subrutinas. La ejecución se realiza por medio de interrupciones.

El movimiento de las piedras y el del personaje principal se hacen en saltos de medio en medio carácter. Para ello se utilizan dos conjuntos de caracteres, con lo cual se consigue una acción aceptablemente suave sin hacer que el programa sea demasiado complicado ni demasiado lento. En todos los juegos de esta clase, la velocidad es importante.



Lo primero que tienes que hacer en cualquier juego es presentar en la pantalla una página con el título. Aunque la rutina de presentación está escrita en código máquina, no tiene mucho sentido el poner en código máquina las palabras que han de aparecer en pantalla. En vez de hacer esto, las palabras que quieras que aparezcan en pantalla se teclean en **ASCII** como parte del siguiente programa en BASIC que las *pokea* en memoria.

Tienes varias formas diferentes de hacer esto. Aquí veremos dos de ellas, por lo que la presentación de la página con el título del juego se divide en dos secciones, cada una de las cuales puede ser ejecutada y probada separadamente.

Antes de introducir ningún programa tienes que desplazar hacia abajo el RAMTOP a fin de definir una zona protegida, lo cual consigues *pokeando* 51 con el valor 255, 52 con 63, 55 con 255 y 56 con 63. A continuación tienes que introducir el programa en BASIC y ejecutarlo. Así se construirá la tabla de datos en la zona protegida de la memoria. Después teclea un NEW para deshacerte del programa *pokeador* en BASIC, carga tu monitor en código máquina y utilízalo para guardar la tabla en cinta.

Teclea nuevamente NEW para librarte del monitor en código máquina y carga tu ensamblador con un LOAD. Teclea la rutina en lenguaje ensamblador y utiliza la opción SAVE para guardar en cinta el código fuente. Seguidamente ensambla la rutina, haz un nuevo NEW para librarte del ensamblador y carga de nuevo el monitor de código máquina. Guárdalo en cinta.

Puedes ejecutar las rutinas en código máquina, llamándolas con SYS 16384. Pero tienes que tener al mismo tiempo en memoria la tabla de datos.

El siguiente programa en BASIC contiene todos los datos de la página

de presentación, excepto la palabra **CLIFF** (abismo). Esta palabra se añade más adelante utilizando un método diferente de entrada de datos.




```
5Ø POKE53281,1
1Ø ADD=1664Ø:FORI=ØTO
   32ØØØ
2Ø READA%:POKEADD+I,A%
25 PRINTCHR$(A%);
3Ø IFA%=ØGOTO5Ø
4Ø NEXT
5Ø END
1664Ø DATA 147,149,142,
      169,169,169,169,169,
      169,169
1665Ø DATA 169,169,169,
      169,169,169,169,169,
      169,169
1666Ø DATA 169,169,169,
      142,13,149,169,169,
      169,169
1667Ø DATA 169,169,169,
      169,169,169,169,169,
      169,169
1668Ø DATA 169,169,169,
      169,169,142,144,125,
      32,32
1669Ø DATA 32,32,32,32,32,
      82,69,86,73,83
1670Ø DATA 84,65,32,13,149,
      169,169,169,169,169
1671Ø DATA 169,169,169,
      169,169,169,169,169,
      169,169
1672Ø DATA 169,169,169,
      144,32,125,32,32,32,
      32
1673Ø DATA 32,32,32,32,73,
      78,8Ø,85,84,46
1674Ø DATA 32,32,32,149,13,
      149,169,169,169,169
1675Ø DATA 169,169,169,
      169,169,169,169,169,
      169,169
1676Ø DATA 169,169,169,
      142,144,32,32,125,32,
      32
1677Ø DATA 32,32,32,32,32,
      32,32,32,32,32
1678Ø DATA 32,32,32,32,32,
      32,13,149,169,169
1679Ø DATA 169,169,169,
      169,169,169,169,169,
      169,169
1680Ø DATA 169,169,169,
      169,144,32,32,32,125,
      32
1681Ø DATA 32,32,32,82,69,
      86,73,83,65,68
1682Ø DATA 79,32,8Ø,79,82,
      13,149,169,169,169
1683Ø DATA 169,169,169,
      169,169,169,169,169,
      169,169
1684Ø DATA 169,169,142,
      144,32,32,32,32,125,
      32
1685Ø DATA 32,32,32,74,79,
      83,69,8Ø,32,77
1686Ø DATA 46,32,71,73,76,
      83,32,32,13,149
1687Ø DATA 169,169,169,
      169,169,169,169,169,
      169,169
1688Ø DATA 169,169,169,
      169,144,32,32,32,32,
      32
1689Ø DATA 125,149,13,169,
      169,169,169,169,169,
      169
1690Ø DATA 169,169,169,
      169,169,169,13,169,
      169,169
1691Ø DATA 169,169,169,
      169,169,169,169,169,
      169,13
1692Ø DATA 169,169,169,
      169,169,169,169,169,
      169,169
1693Ø DATA 169,13,169,169,
      169,169,169,169,169,
      169
1694Ø DATA 169,169,13,169,
      169,169,169,169,169,
      169
1695Ø DATA 169,169,13,169,
      169,169,169,169,169,
      169
1696Ø DATA 169,13,169,169,
      169,169,169,169,169,
      13
1697Ø DATA 169,169,169,
      169,169,169,13,169,
      169,169
```

```
16980 DATA 169,169,13,169,
      169,169,169,13,169,
      169
16990 DATA 169,13,169,169,
      13,169,13,32,32,32
17000 DATA 32,32,32,32,32,
      142,31,178,32,178
17010 DATA 32,117,99,105,
      32,117,105,178,32,117
17020 DATA 99,105,32,117,
      99,105,32,176,99,105
17030 DATA 13,32,32,32,32,
      32,32,32,32,125
17040 DATA 32,125,32,125,
      32,125,32,125,125,125
17050 DATA 32,125,32,125,
      32,125,32,32,32,125
17060 DATA 32,125,13,32,32,
      32,32,32,32,32
17070 DATA 32,171,99,179,
      32,171,99,179,32,125
17080 DATA 125,125,32,125,
      32,32,32,171,179,32
17090 DATA 32,171,178,107,
      13,32,32,32,32,32
17100 DATA 32,32,32,125,32,
      125,32,125,32,125
17110 DATA 32,125,125,125,
      32,125,32,178,32,125
17120 DATA 32,32,32,125,
      125,13,32,32,32,32
17130 DATA 32,32,32,32,177,
      32,177,32,177,32
17140 DATA 177,32,177,106,
      107,32,106,99,107,32
17150 DATA 106,99,107,32,
      177,202,203,0
```

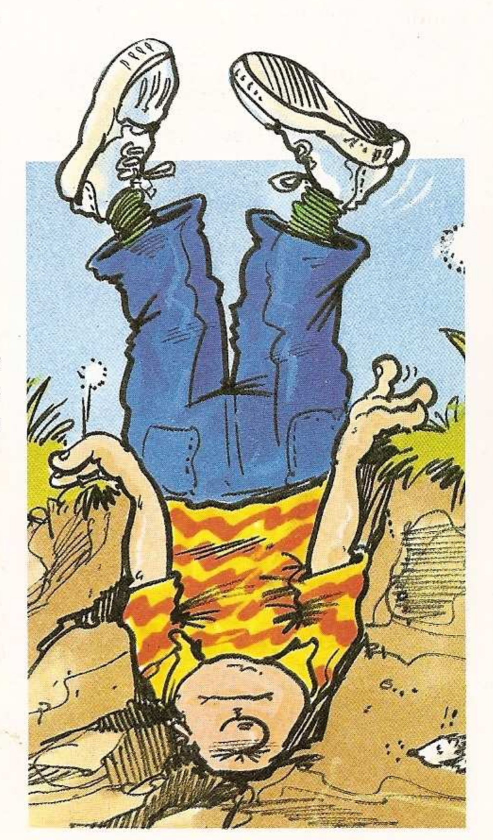
La tabla de datos creada al ejecutar el programa anterior es leída con la siguiente rutina en código máquina, que controla la presentación en pantalla:

```
ORG 16384
NOP
NOP
NOP
LDA #$09
STA $D020
LDA #$03
STA $D021
LDA #$00
STA $FB
LDA #$41
STA $FC
LDY #$00
NOP
LDA ($FB),Y
NOP
NOP
NOP
NOP
BEQ $402A
JSR $FFD2
NOP
INC $FB
BNE $R026
INC $FC
CLC
BCC $4018
NOP
NOP
NOP
NOP
NOP
NOP
NOP
NOP
JSR $FFE4
BEQ $4035
RTS
```

EL BASIC

El programa BASIC utiliza un sencillo bucle FOR ... NEXT para leer los DATA mediante sentencias READ y *pokearlos* en memoria en una tabla a la cual pueda acceder el programa en código máquina.

La tabla de datos empieza en la dirección 16640, pero es evidente que no contiene 32000 datos, que es el valor de I que controla el bucle FOR ... NEXT. La línea 30 detiene el programa cuando hay un cero en los datos. El resto de los datos son los códigos ASCII de los caracteres, los símbolos gráficos de Commodore y los códigos de control. Puedes encontrar todos estos códigos en el *Apéndice F de la Guía de Usuario del Commodore 64* o en el *Apéndice C de la Guía de Referencia para Programadores.*



PRESENTACIONES EN PANTALLA

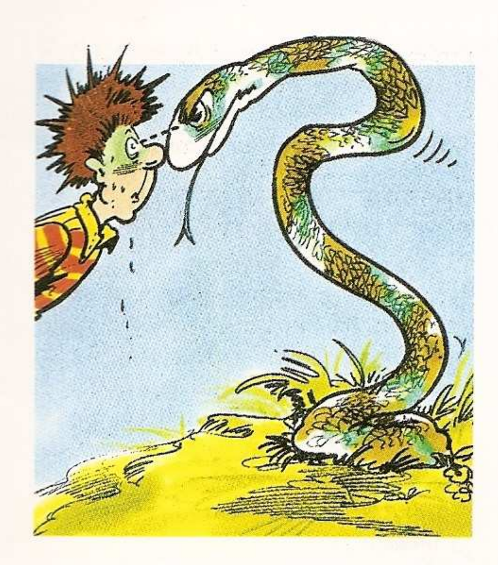
Como sabes, NOP significa No Operación, lo cual quiere decir que esta instrucción no hace nada. Sin embargo, no es absolutamente inútil. Se suele emplear para disminuir la velocidad del microprocesador. Cuando se pone dentro de un bucle se queda sin hacer nada una y otra vez. Pero aquí la utilizamos como herramienta de programación.

Las instrucciones NOP se utilizan aquí para cortar los programas de forma que puedas ver claramente lo que está sucediendo. Con esto también se le ofrece al programador la posibilidad de añadir una instrucción adicional, en el supuesto de que la necesite, dejándole bytes disponibles para almacenamientos temporales cuando haga falta.

La primera instrucción activa carga el acumulador con el valor nueve y a continuación lo almacena en la dirección de memoria D020. Esta dirección se encuentra en la zona de entradas/salidas y controla el color del borde de pantalla. Responde a los mismos números de colores que se utilizan en el BASIC. Con el valor 9 se obtiene un borde de color *marrón*. Seguidamente se almacena el valor 3 en la dirección D021, con lo que resulta una pantalla de color *cyan*.

A continuación se carga la dirección de comienzo de la tabla de datos, que es 16640, en las direcciones FB y FC de la página cero; recuerda que 16640 en decimal es 4100 en hexadecimal. Después se pone a cero el registro Y.

La instrucción LDA ($FB),Y carga en el acumulador el primer byte de la tabla de datos. Observa que el desplazamiento Y sigue siendo cero durante toda la rutina de presentación en pantalla mientras que se va actualizando el puntero de la tabla de datos, contenido en FB y FC. Pero aquí necesitamos un direccionamiento indirecto, cosa que en el 6520 sólo está disponible en forma indexada.



La instrucción BEQ $402A saca al microprocesador de la rutina cuando se llega al cero que hay al final de la tabla de datos. A continuación JSR $FFD2 fuerza el salto a la subrutina del Kernal ROM, encargada de presentar un carácter en pantalla. Fíjate en que no tienes que decirle dónde tiene que imprimirlo. Con este método, el cursor se desplaza a la posición correcta de impresión, partiendo de los códigos de control contenidos en los datos.

INC $FB incrementa ahora el byte bajo del puntero. Si el resultado no es cero, la instrucción BNE fuerza un salto por encima de la instrucción siguiente, la cual incrementa el byte alto cuando se llega al final de una página.

El bucle se cierra con las instrucciones CLC y BCC $4018. Después de una instrucción CLC, el carry siempre estará a cero, por lo que siempre se cumplirá la condición para que se ejecute la instrucción BCC y se producirá un salto hacia atrás hasta LDA ($FB), Y cargándose el siguiente byte de la tabla de datos.

EL BUCLE DE DEPURACION

En cuanto se ha presentado en la pantalla el último carácter de la tabla de datos, y se carga el delimitador de cero, el microprocesador salta hacia afuera de la rutina. Pero no te va a interesar volver inmediatamente al BASIC, ya que en tal caso los mensajes no estarían en pantalla el tiempo suficiente para que puedas comprobar si el programa funciona adecuadamente o no. Por eso hemos incorporado un bucle de depuración.

La instrucción JSR $FFE4 fuerza un salto a la subrutina del Kernal ROM encargada de detectar las pulsaciones de tecla. Si ha habido alguna pulsación, regresa al acumulador con el valor de la tecla pulsada. Al poner dicho valor en el acumulador, activa los indicadores.

Si no se ha pulsado ninguna tecla, el valor resultante es 0 y la instrucción BEQ $4035 fuerza un retroceso a JSR $FFE4. Pero si en cambio se pulsó alguna tecla, el acumulador contendrá un valor distinto de cero y no se activará el indicador de cero. En consecuencia la instrucción BEQ no forzará salto alguno y el micro abandonará la rutina.

En otras palabras, el microprocesador recorre esta rutina una vez y otra, manteniendo en la pantalla los titulares de presentación del juego, hasta que se pulse una tecla cualquiera.

La siguiente rutina escribirá por encima de este bucle de depuración, pero puedes usarlo para ir comprobando lo que llevas tecleado.

EL ACANTILADO

Ahora tienes que teclear el siguiente programa de pokeo en BASIC, ejecutarlo, y guardar en cinta mediante un SAVE la tabla de datos que construye, igual que hicimos antes:

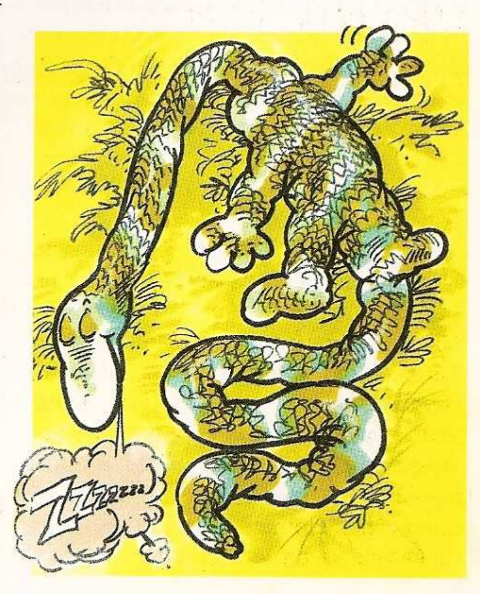
```
10 ADD=17184:FORI=0TO
   32000
20 READA%:POKEADD+I,A%
30 IFA%=255GOTO50
40 NEXT
50 END
17184 DATA 8,21,31,117,105,
      0,9,21,98,0
17194 DATA 10,21,106,107,
      178,0,11,23,98,0
17204 DATA 12,23,173,189,
      178,0,13,25,98,0
17214 DATA 14,25,177,176,
      174,0,15,26,171,0
17224 DATA 16,26,177,176,
      174,0,17,27,171,0
17234 DATA 18,27,177,0,0,0,
      255,0
```

A continuación puedes teclear la rutina en código máquina, ensamblarla, guardarla y llamarla, utilizando nuevamente el método visto anteriormente:

```
ORG 16437
LDY #$FF
INY
LDX $4320,Y
INY
LDA $4320,Y
BEQ $4059
INY
STY $FB
TAY
JSR $FFF0
LDY $FB
LDA $4320,Y
BEQ $4037
INY
JSR $FFD2
CLC
BCC $404A
NOP
NOP
NOP
JSR $FFE4
BEQ $4059
```

FUNCIONAMIENTO DEL PROGRAMA

Esta parte del programa utiliza los símbolos gráficos del Commodore para formar letras ampliadas. Dichas letras se presentan directamente en la pantalla utilizando la rutina PLOT del Kernal ROM. Esta rutina sitúa el cursor en una posición especificada de la pantalla, imprimiendo a continuación el carácter en dicha posición.



En el programa de BASIC, cada sección de los DATA está delimitada por un 0. Los dos primeros valores de cada sección especifican las coordenadas X e Y del principio de una línea de caracteres. La propia tabla finaliza con tres ceros.

El 255 que figura a continuación sirve para informar al programa de pokeado de cuándo tiene que pararse. El propio programa pokeador empieza por construir su tabla de datos partiendo de 17184 hasta completar los datos de la página de presentación. Pero es evidente que no hay 32000 valores, como implicaría el límite del valor de I en el bucle FOR ... NEXT. La línea 30 se ocupa de esto, detectando el momento en que se alcanza el valor 255 y dando fin al programa.

EL CODIGO MAQUINA

La rutina en código máquina comienza inicializando un índice en el registro Y. Se fija su valor en FF, ya que Y se incrementa al principio del bucle. De esta forma, cuando el procesador se va por primera vez a la rutina de impresión, el valor de Y es cero. En esta ocasión utilizamos un índice, a diferencia de la primera parte de la rutina de impresión en que utilizábamos un puntero de dos bytes con índice nulo, debido a que los datos no van a superar los 255 bytes.

LDX $4320,Y carga en el registro Y el primer byte de la tabla de datos, es decir, la coordenada Y del comienzo de la primera línea de caracteres. Se incrementa el registro Y y se carga el segundo byte de la tabla, la coordenada X, en el acumulador. El valor 4320 en hexadecimal es 17184 en decimal, que es la dirección inicial del principio de la tabla de datos.

Si esta segunda coordenada es cero, la instrucción BEQ $4059 fuerza la salida de la rutina. Por eso se utilizan tres ceros para marcar el final de los datos. El primero de ellos indica al procesador que debe volver hacia atrás y cargar nuevas coordenadas correspondientes al principio de la línea siguiente. A continuación el índice Y cuenta dos elementos más de la tabla hasta que se ensaya un byte.

El índice Y se incrementa y se almacena en la dirección FB de la página cero, ya que se necesitará más adelante. La instrucción TAY transfiere al registro Y la coordenada contenida en el acumulador. Ahora ya están las coordenadas en los registros requeridos por la rutina PLOT.

La instrucción JSR $FFF0 fuerza un salto a la rutina ROM que mueve el cursor a la posición especificada en los registros X e Y.

Cuando se posiciona el cursor y el procesador regresa a esta rutina, se restaura de nuevo el registro Y, cargándolo a partir de FB.

Después LDA $4320,Y carga en el acumulador el siguiente byte de la tabla de datos; Y había sido ya incrementado antes de transferirlo. Si el byte cargado era un cero, la instrucción BEQ $4037 fuerza un salto hacia atrás hasta la primera instrucción INY, quedando en condiciones de cargar las coordenadas iniciales de la línea siguiente o salir de la rutina.

Si el byte cargado no es un cero, el índice Y se incrementa nuevamente y el procesador salta a la subrutina de ROM encargada de sacar los caracteres por pantalla. CLC y BCC $404A envían al procesador a cargar y presentar el siguiente byte de la tabla de datos.

Al leer la línea DATA, observarás que 8 y 21 son las coordenadas Y y X de la primera posición de impresión. El número 31 hace que los siguientes caracteres aparezcan en azul. 117 y 105 son los dos arcos que juntos configuran la parte superior de la letra C.

La siguiente línea empieza con 9 y 21, que es un carácter cuadrado situado debajo del principio de la línea anterior. El 98 da una línea vertical que forma la parte posterior de la C, y así sucesivamente. Puedes ir deduciendo los caracteres a imprimir a partir de las tablas del *Apéndice F de la Guía de Usuario del Commodore 64*, o del *Apéndice C de la Guía del Programador*.

Observando las coordenadas puedes ver que las letras se imprimen en la pantalla, desplazadas todas ellas un carácter cuadrado hacia la derecha. De aquí resulta el declive de la palabra CLIFF por la pantalla.

CORRECCIONES

Normalmente, el procesador se iría desde aquí al resto del programa, pero por ahora hemos llegado al final de la primera parte de *Abismo*. Por eso necesitas de momento otro bucle de depuración que retenga la presentación en pantalla a fin de que puedas ver si el programa trabaja de manera adecuada.

Los lectores que deseéis entrar este programa debéis seguir los siguientes pasos:

1) Limpiar la zona de trabajo y/o leer rutinas ya grabadas.
2) Entrar las diferentes subrutinas del artículo.
3) Guardar la nueva versión del programa

PRIMER PASO: Limpiar la zona de trabajo.

a) Desde un ensamblador o monitor F3ØØØ, 69ØØ,ØØ
b) Desde BASIC hacer en modo directo:
FOR N=12288 TO 2688Ø: POKE N,Ø: NEXT (RETURN)

LECTURA DE RUTINAS GRABADAS:

a) Desde ensamblador L "nombre del programa",xx
b) Desde BASIC hacer LOAD "nombre del programa",xx,1 Siendo xx=Ø1 para la cinta y Ø8 para el disco

SEGUNDO PASO: Entrada de rutinas

a) Desde ensamblador: entrar las rutinas en ensamblador, o bien salir hasta el BASIC y entrar los cargadores de BASIC
b) Desde el BASIC: entrar los cargadores de BASIC

TERCER PASO: Guardar el programa

a) Desde ensamblador: S"nombre nueva version",xx
b) Desde el BASIC hacer en modo directo:
POKE 43,Ø: POKE 44,48 (POKE 45,Ø: POKE 46,1Ø5 (return)
A continuación hacer SAVE" nombre nueva version",xx,1

ARRANQUE DEL PROGRAMA:

Hacer SYS 26448 cuando estén instaladas todas las subrutinas.

PROGRAMA MULTIGESTION

Este mes, en esta sección, os presentamos un programa que engloba cinco utilidades: un fichero capaz de almacenar la información de 100 fichas; una pequeña guía telefónica; un programa de contabilidad; un control de stocks y un último programa que nos permite construir gráficos de barras. Es lo que se llama un programa integrado.

Los subprogramas de control de stocks, la guía telefónica y la contabilidad ya se encuentran explicados en sus propios menús y son de fácil manejo. Las instrucciones para trabajar con el fichero son:

e: para entrar datos en las fichas.
s: para pasar a la siguiente ficha.
a: para acceder a la ficha anterior.
n: para buscar una ficha por su número.
h: para buscar una ficha por el índice.

«Carga», «salvado», «salida» son instrucciones del fichero que debemos escribir por completo. «Salvado» graba la cinta en el contenido de las fichas. «Carga» recupera de la cinta dicho contenido. Y «salida» vuelve el programa al menú principal. Para utilizar el programa de gráficos de barras tan sólo debemos saber que «s» es para volver al menú principal y «o» para pasar al siguiente gráfico.

Para acceder al programa se nos pedirá un código, y éste es: S00e o XSPe S00ex0a.

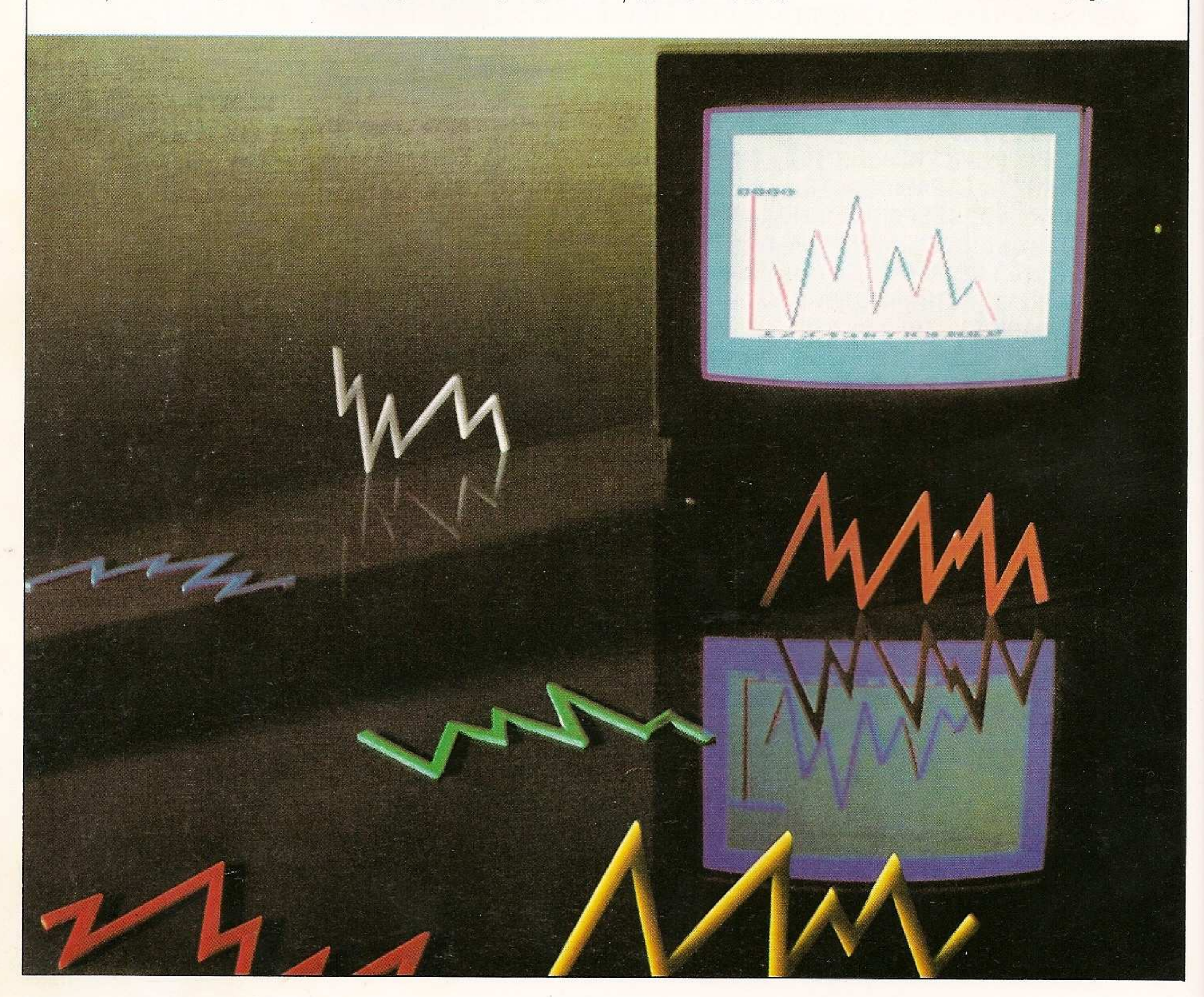
FÉLIX ORTEGA

```
10 A$(1)="GASTOSDOMESTICOS":A$(2)="ENTRETENIMIENTO": A$ (3)="ALQUILERES"
15 A$(8)="INGRES.":A$(4)="ROPA":A$(5)="COCHE":A$(6)="VACACION":A$(7)="VARIOS"
2Ø DIMC$(100,8):C=1:DIMS$(300,4):DIMD(15):DIMT$(1ØØ,1)
25 DIMD$(4,4ØØ):CO=Ø:GOTO 86Ø
3Ø REM ***** CONTROL DE STOCKS *****
35 POKE 5328Ø,Ø:POKE 53281,Ø:PRINT"[SHIFT+CLR /HOME][CTRL+6][ 10*CRSR ABAJO][
   8*CRSR DCHA.]";
4Ø PRINT "[SHIFT+C]ONTROL DE [SHIFT+S] TOCKS": FOR A=Ø TO 9ØØ:NEXT A
45 PRINT "[SHIFT+CLR/HOME]";TAB(15); "[SHIFT+O]PCIONES": PRINT TAB(15);
   "========": PRINT "[ 2*CRSR ABAJO]"
5Ø PRINT "[SHIFT+C]ARGAR EXISTENCIAS ACTUALES [ 2 ESPACIOS] ========> 1"
55 PRINT "[SHIFT+S]ALVAR EXISTENCIAS ACTUALES [ 2 ESPACIOS] ========> 2"
6Ø PRINT "[SHIFT+L]ISTAR EXISTENCIAS ACTUALES [ 2 ESPACIOS] ========> 3"
65 PRINT "[SHIFT+L]ISTAR EXISTENCIAS A RENOVAR ========> 4"
7Ø PRINT "[SHIFT+C]AMBIAR DATOS DE UN ELEMENTO ========> 5"
75 PRINT "[SHIFT+E]NTRAR NUMERO DE ELEMENTOS VENDIDOS => 6"
8Ø PRINT "[SHIFT+S]ALIDA AL MENU PRINCIPAL [ 5 ESPACIOS] ========> 7"
85 PRINT "[ 3*CRSR ABAJO] [ 4Ø*COMM+@]"
9Ø PRINT" ↑ [3]=>[SHIFT+M]ENU/[2ESPACIOS][CTRL+9][SHIFT+P]RES.OPCIONDESEADA
   [CTRL+Ø]";
95 GET A$
1ØØ IF A$< "1" OR A$>"7" THEN 95
1Ø5 PRINT "[SHIFT+CLR/HOME]": ON VAL (A$) GOSUB 153Ø,144Ø,115,18Ø,24Ø,39Ø,55Ø
11Ø GOTO 45
115 FOR A=Ø TO 3ØØ
12Ø IF S$ (A,Ø) = " " THEN 17Ø
125 PRINT "[SHIFT+CLR/HOME]"
13Ø PRINT "[SHIFT+L]IBRO:";S$(A,Ø)
135 PRINT "[SHIFT+E]DITORIAL: ";S$(A,1)
```

```
140 PRINT "[SHIFT+A]UTOR: ";S$(A,2)
145 PRINT "[SHIFT+C]ODIGO REF.:" ;S$(A,3)
150 IF VAL(S$(A,4))<VAL(S$(A,3)) THEN PRINT "[CTRL +9] [SHIFT+E]XISTENCIAS
    RESTANTES:";S$(A,4);"[CTRL+0]"
155 IFVAL(S$(A,4))>=VAL(S$(A,3))THENPRINT"[SHIFT+E]XISTENCIASRESTANTES:";S$(A,4)
160 GET A$: IF A$ = "  " THEN 160
165 IF A$ = "↑" THEN 45
170 NEXT A
175 RETURN
180 OPEN 4,4:CMD 4
185 FOR A=0 TO 300
190 IF S$(A,0)= "  " THEN 230
195 IF VAL (S$ (A, 4))> VAL(S$(A,3)) THEN 230
200 PRINT "[SHIFT+L]IBRO:" ;S$(A,0)
205 PRINT "[SHIFT+E]DITORIAL:" ;S$(A,1)
210 PRINT "[SHIFT+A]UTOR:" ;S$(A,2)
215 PRINT "[SHIFT+C]ODIGO REF.:" ;S$(A,3)
220 PRINT "[SHIFT+E]XISTENCIAS RESTANTES:" ;S$(A,4)
225 PRINT "***************************************"
230 NEXT A
235 CLOSE 4:RETURN
240 OPEN 1,0,0: PRINT "[SHIFT+N]OMBRE DEL LIBRO": INPUT#1,A$: PRINT
245 D=-1
250 FOR A=0 TO 300
255 IF S$(A,0)=A$ AND D=-1 THEN D=A
260 NEXT A
265 IF D=-1 THEN PRINT "[SHIFT+N]O SE HA ENCONTRADO ESE LIBRO.": CLOSE 1:GOTO 240
270 A=D
275 PRINT "[SHIFT+N]OMBRE DEL LIBRO:" ;S$(A,0)
280 PRINT "[SHIFT+E]DITORIAL:" ;S$(A,1)
285 PRINT "[SHIFT+A]UTOR:" ;S$(A,2)
290 PRINT "[SHIFT+C]ODIGO REF.:" ;S$(A,3)
295 PRINT "[SHIFT+E]XISTENCIAS RESTANTES" ;S$(A,4)
300 PRINT " ***************************************"
305 PRINT "[SHIFT+N]UEVO NOMBRE DEL LIBRO:": INPUT#1,A$:PRINT
310 IF A$="  " THEN A$=S$(A,0)
315 S$(A, 0)=A$
320 PRINT "[SHIFT+N]UEVA EDITORIAL:" ;: INPUT#1,A$:PRINT
325 IF A$="  " THEN A$=S$(A,1)
330 S$(A,1)=A$
335 PRINT "[SHIFT+N]UEVO AUTOR:" ;:INPUT#1,A$:PRINT
340 IF A$="  " THEN A$=S$(A,2)
345 S$(A,2)=A$
350 PRINT "[SHIFT+N]UEVO CODIGO REF.:" ;:INPUT#1,A$:PRINT
355 IF A$="  " THEN A$=S$(A,3)
360 S$(A,3)=A$
365 PRINT "[SHIFT+N]UEVAS EXISTENCIAS RESTANTES:" ;:INPUT#1,A$:PRINT
370 IF A$="  "THEN A$=S$(A,4)
375 S$(A,4)=A$
380 CLOSE 1
385 RETURN
```

```
390 OPEN 1,0,0: PRINT "[SHIFT+N]OMBRE DEL LIBRO": INPUT#1,A$:PRINT
395 D=-1
400 FOR A=0 TO 300
405 IF S$(A,0)=A$ THEN D=A
410 NEXT A
415 IF D=-1 THEN PRINT "[SHIFT+N]O HAY TAL LIBRO":CLOSE 1:GOTO 390
420 A = D
425 PRINT "[SHIFT+E]XISTENCIAS ANTERIORES:" ;S$(A,4)
430 PRINT "[SHIFT+N]UMERO VENDIDO:" :INPUT#1, N:CLOSE 1
435 S$ (A,4)=STR$(VAL(S$(A,4))-N)
440 RETURN
445 POKE 53280,0:POKE 53281,0:PRINT CHR$(8);CHR$(14);"[SHIFT+CLR/HOME][2*CRSR
    ABAJO][4*CRSRDCHA.][CTRL+6]";
450 PRINT"[CTRL+9][5ESPACIOS][CTRL+0][2ESPACIOS][CTRL+9][5ESPACIOS][CTRL+0][2
    ESPACIOS][SHIFT+ESPACIO][CTRL+9][4ESPACIOS][CTRL+0][3ESPACIOS][CTRL+9][3
    ESPACIOS][CTRL+0][3ESPACIOS][CTRL+9][2ESPACIOS][CTRL+0][CTRL+9][2ESPACIOS]
    [CTRL+0]"
455 PRINT"[4ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS]
    [CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0][2
    ESPACIOS][CTRL+9][CTRL+0][CTRL+9][CTRL+0][CTRL+9][CTRL+0]"
460 PRINT"[4ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS]
    [CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0][2
    ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0]"
465 PRINT"[4ESPACIOS][CTRL+9][3ESPACIOS][CTRL+0][4ESPACIOS][CTRL+9][3ESPACIOS]
    [CTRL+0][4ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS]
    [CTRL+9][CTRL+0][2ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0]"
470 PRINT"[4ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS]
    [CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0][2
    ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0]"
475 PRINT"[4ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS]
    [CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0][2
    ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0]"
480 PRINT"[4ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][5ESPACIOS][CTRL+0][3
    ESPACIOS][CTRL+9][4ESPACIOS][CTRL+0][3ESPACIOS][CTRL+9][3ESPACIOS][CTRL+0][3
    ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS][CTRL+9][CTRL+0]"
485 PRINT"[3*CRSRABAJO][12ESPACIOS][CTRL+9][4ESPACIOS][CTRL+0][2ESPACIOS][CTRL+
    9][5ESPACIOS][CTRL+0][5ESPACIOS][CTRL+9][CTRL+0]"
490 PRINT"[11ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][8ESPACIOS]
    [CTRL+9][2ESPACIOS][CTRL+0]"
495 PRINT"[11ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][7ESPACIOS]
    [CTRL+9][CTRL+0][CTRL+9][CTRL+0]"
500 PRINT"[11ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][5ESPACIOS][CTRL+0][2
    ESPACIOS][CTRL+9][CTRL+0][2ESPACIOS][CTRL+9][CTRL+0]"
505 PRINT"[11ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS]
    [CTRL+9][CTRL+0][2ESPACIOS][CTRL+9][5ESPACIOS][CTRL+0]"
510 PRINT"[11ESPACIOS][CTRL+9][CTRL+0][6ESPACIOS][CTRL+9][CTRL+0][3ESPACIOS]
    [CTRL+9][CTRL+0][5ESPACIOS][CTRL+9][CTRL+0]"
515 PRINT"[12ESPACIOS][CTRL+9][4ESPACIOS][CTRL+0][SHIFT+ESPACIO][CTRL+9][5
    ESPACIOS][CTRL+0][SHIFT+ESPACIO][4ESPACIOS][CTRL+9][CTRL+0]"
520 PRINT "[ 3*CRSR ABAJO] [ 40*COMM+@]";
```

```
525 PRINT"[SHIFT+R]ETURN=>AVANZAR[9ESPACIOS]F1=>[CTRL+9][SHIFT+T]ERMINAR
    [CTRL+Ø]";
53Ø GET A$:IF A$="  " THEN 53Ø
535 IF A$=CHR$(13) THEN 55Ø
54Ø IF A$=CHR$(133) THENPRINT "[SHIFT+CLR/HOME]": SYS 58648: END
545 IF A$<>CHR$(13) OR A$<>CHR$(133) THEN 53Ø
55Ø REM ***** MENU PRINCIPAL *****
555 POKE53272,23:POKE56325,58:PRINT"[SHIFT+CLR/HOME][CRSRABAJO][13*CRSRDCHA.]
    [SHIFT+M]ENUPRINCIPAL"
56Ø PRINT "[ 13 * CRSR DCHA.] ==============="
565 PRINT"[2*CRSRABAJO][CTRL+9]F1[CTRL+Ø][2ESPACIOS]===============>[2
    ESPACIOS][SHIFT+F]ICHERO "
57Ø PRINT"[2*CRSRABAJO][CTRL+9]F2[CTRL+Ø][2ESPACIOS]===============>[2
    ESPACIOS][SHIFT+L]ISTINTELEFONICO "
575 PRINT"[2*CRSRABAJO][CTRL+9]F3[CTRL+Ø][2ESPACIOS]===============>[2
    ESPACIOS][SHIFT+C]ONTABILIDAD "
58Ø PRINT"[2*CRSRABAJO][CTRL+9]F5[CTRL+Ø][2ESPACIOS]===============>[2
```

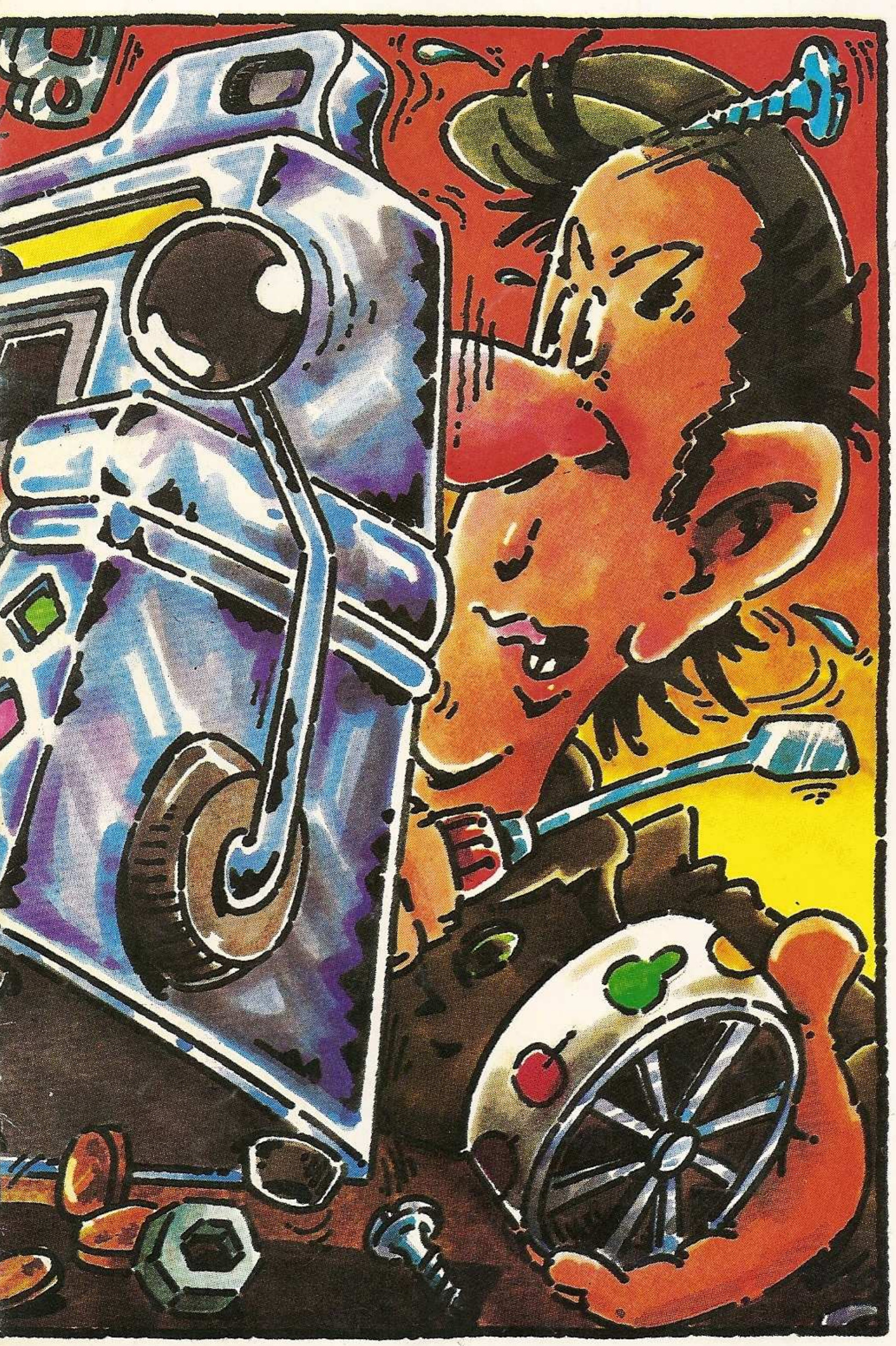


```
     ESPACIOS][SHIFT+S]TOCKS[3ESPACIOS]"
585 PRINT"[2*CRSRABAJO]F7[2ESPACIOS]===============>[2ESPACIOS][SHIFT+
     G]RAFICOS"
59Ø PRINT"[2*CRSRABAJO][10ESPACIOS][CTRL+9][SHIFT+P]ULSEOPCIONDESEADA[CTRL+
     Ø]"
595 PRINT "[CRSR ABAJO] [ 40*COMM+@]";
6ØØ PRINT"[SHIFT+H]=>HORA[11ESPACIOS][SHIFT+R]ETURN=>[SHIFT+F][SHIFT+E][SHIFT
     +C][SHIFT+O][SHIFT+M][2*SHIFT+ESPACIO][SHIFT+C]64";
6Ø5 GET N$:IF N$="" THEN 6Ø5
61Ø IF N$="H" OR N$="[SHIFT+H]" THEN 1135
615 IF N$=CHR$(13) THEN 445
62Ø IF N$="[F2]" THEN 116Ø
625 IF N$=CHR$(133) THEN 65Ø
63Ø IF N$=CHR$(134) THEN 163Ø
635 IF N$=CHR$(136) THEN 95Ø
64Ø IF N$=CHR$(135) THEN 3Ø
645 IF N$<>CHR$(13) OR N$<> CHR$(133) OR N$<>CHR$(134) OR N$<>CHR$(135) THEN
     925
65Ø REM ***** FICHERO *****
655 PRINT "[SHIFT+CLR/HOME]"
66Ø PRINT "[COMM+A][ 38*SHIFT+*][COMM+S]";
665 FOR E=Ø TO 16
67Ø PRINT "[SHIFT+-][ 38 ESPACIOS][SHIFT+-]";
675 NEXT E
68Ø PRINT "[COMM+Z][ 38*SHIFT+*][COMM+X]"
685 PRINT "[CLR/HOME][ 2*CRSR ABAJO][CRSR DCHA.][SHIFT+F]ICHA ";C,"=>";C$ (C,Ø)
69Ø FOR E=1 TO 8: PRINT"[CRSR ABAJO][CRSR DCHA.]";C$(C,E): NEXT E
695 PRINT "[ 3*CRSR ABAJO] =>;:OPEN 1,Ø, Ø: INPUT#1,D$:PRINT: CLOSE 1
7ØØ IF D$="E" OR D$=[SHIFT+E] THEN GOSUB 745
7Ø5 IF D$="CARGA" THEN GOSUB 153Ø
71Ø IF D$="SALVADO" THEN GOSUB 144Ø
715 IF D$="S" THEN C=C+1: IF C>1ØØ THEN C=1ØØ
72Ø IF D$="A" THEN C=C-1: IF C<=Ø THEN C=1
725 IF D$="N" THEN INPUT C:IFC<=Ø OR C> 1ØØ THEN C=1
73Ø IF D$="B" THEN GOSUB 765
735 IF D$="SALIDA" THEN 55Ø
74Ø PRINT "[SHIFT+CLR/HOME]": GOTO 66Ø
745 REM ***** ENTRADA *****
75Ø PRINT "[CLR/HOME][ 2*CRSR ABAJO]",; "[ 3*CRSR DCHA.]";:GOSUB 8ØØ: C$(C,
     Ø)=Q$:PRINT
755 FORE=1TO8:PRINT"[CRSRABAJO][CRSRDCHA.]";:GOSUB8ØØ:C$(C,E)=Q$:PRINT:NEXTE
76Ø RETURN
765 INPUT "[SHIFT+T]ITULO=>";T$
77Ø C1 = - 1
775 FOR F=Ø TO 1ØØ
78Ø IF C$(F,Ø)=T$ THEN C1=F
785 NEXT F
79Ø IF C1=-1 THEN PRINT "[SHIFT+N]O ESTA ALMACENADO TAL TITULO": GOTO 765
795 C=C1: RETURN
8ØØ Q$=" "
```

```
8Ø5 GET K$
81Ø IF K$=CHR$(17) OR K$=CHR$(145) THEN 8Ø5
815 IF K$=CHR$(13) THEN PRINT "[SHIFT+ESPACIO]";: RETURN
82Ø IF K$=CHR$(29) THEN PRINT"[CTRL+Ø] [CRSR IZQDA.]";
825 IF K$=CHR$(157) THEN PRINT "[CTRL+Ø] [CRSR IZQDA.]";
83Ø IF K$=CHR$(2Ø) OR K$=CHR$(148) THEN 8Ø5
835 IF K$=CHR$(19) OR K$=CHR$(147) THEN 8Ø5
84Ø IF K$=CHR$(34) THEN8Ø5
845 PRINT K$;"[CTRL+9][CTRL+Ø][CRSR IZQDA.]";
85Ø Q$=Q$+K$
855 GOTO 8Ø5
86Ø REM ***** ENTRADA SISTEMA *****
865 POKE65Ø,128:POKE5328Ø,Ø:POKE53281,Ø:PRINT"[SHIFT+CLR/HOME]";CHR$(8);CHR$(14)
87Ø PRINT "[CTRL+6][SHIFT+CLR/HOME][ 23*CRSR ABAJO][ 40*COMM+@]";
875 PRINT "[SHIFT+C]ODIGO S. POR FAVOR => [CTRL+1]";
88Ø INPUT CS$
885 IF CS$="XSPE SØØEXØA" OR CS$="SØØE" THEN 9ØØ
89Ø IF CS$<>"XSPE SØØEXØA" OR CS$<>"SØØE" OR C$="  " THEN 895
895 POKE 53272 4Ø: POKE 56325,58: POKE 648,128: STOP
9ØØ PRINT"[CTRL+6][CRSRABAJO][SHIFT+CLR/HOME][22*CRSRABAJO][4Ø*COMM+@]";
9Ø5 PRINT "[CTRL+9][3 ESPACIOS][SHIFT+C]ODIGO ACEPTADO –[2
    ESPACIOS][SHIFT+E]NTRADA INICIADA [2 ESPACIOS][CTRL+Ø]";
91Ø FOR WW=Ø TO 1ØØØ
915 NEXT WW
92Ø GOTO 11Ø5
925 REM ***** OPCION NO VALIDA *****
93Ø PRINT "[CLR/HOME][ 22*CRSR ABAJO] [ 4Ø *COMM+@]";
935 PRINT "  ********** [SHIFT+O]PCION NO VALIDA **********  ";
94Ø FOR WW=Ø TO 1ØØØ: NEXT
945 POKE 56325,255: GOTO 55Ø
95Ø REM ***** GRAFICOS *****
955 PRINT "[SHIFT+CLR/HOME]"
96Ø INPUT "[SHIFT+R]OTULO VERTICAL";V$
965 INPUT "[SHIFT+R]OTULO HORIZONTAL ";H$
97Ø PRINT: FOR A=1 TO 15
975 PRINT "[SHIFT+V]ALOR DEL DADO ";A;: INPUT D(A)
98Ø NEXT A
985 M=Ø
99Ø FOR A=1 TO 15
995 IF D(A)>M THEN M=D(A)
1ØØØ NEXT A
1ØØ5 S=2Ø/M
1Ø1Ø FOR A=1 TO 15
1Ø15 D(A)=D(A)*S
1Ø2Ø NEXT A
1Ø25 REM ***** DIBUJO GRAFICO *****
1Ø3Ø PRINT "[SHIFT+CLR/HOME]": FOR A=1 TO LEN(V$)
```

Debido a la extensión de este programa su segunda y última parte se publicará en el siguiente número de INPUT.

```
     DCHA.][COMM.+1]
     [SHIFT+P][SHIFT+E]
     [SHIFT+I][SHIFT+G]
     [SHIFT+A][CLR/HOME]";
     :GOTO 3Ø2Ø
3Ø65 H%=-1:I%=-1:
     J%=-1:GOTO 3Ø2Ø
45ØØ FOR A=176Ø TO 1772
     STEP3:FOR B=A TO
     A+1:POKE B+54272,
     9:NEXT B
45Ø5 GET A$:IF A$=CHR$(32)
     THENN%=(A-1757)/
     3:GOTO 4515
451Ø FOR B=A TO A+1:POKE
     B+54272,1Ø:NEXT B,
     A:GOTO 45ØØ
4515 FOR Z=N% TO 1
     STEP-1:KK=((Z-1)*3)+
     176Ø:POKE KK+54272,
     9:POKE KK+
     54273,9
452Ø GET A$:IF A$>"9" OR
     (A$<"5" AND A$<>"Ø")
     THEN452Ø
4525 IF A$="Ø" THENA$=
     "1Ø"
453Ø ON VAL(A$)-4GOSUB
     5ØØØ,6ØØØ,7ØØØ,55ØØ,
     65ØØ,75ØØ
4535 DR=5Ø:DM=4Ø:GOSUB
     8ØØØ:GOSUB
     2ØØØ:KK=((Z-1)*3)+
     176Ø
454Ø IF D%=Ø AND Z=1
     THEN455Ø
4545 IF D%=Ø THENPOKE
     KK+54272,1Ø:POKE
     KK+54273,1Ø:
     NEXT Z
455Ø FOR A=176Ø TO 1772
     STEP3:FOR B=A TO
     A+1:POKE B+54272,
     1Ø:NEXT B,A:
     RETURN
5ØØØ PRINT "[CLR/HOME][
     4*CRSR ABAJO][ 8*CRSR
     DCHA.]";F$(R1%(P%));"[
     2*CRSR ABAJO][ 3*CRSR
     IZQDA.]";
5ØØ5 IF P%=15 THENP%=
     -1
5Ø1Ø P%=P%+1:PRINT
     F$(R1%(P%));"[ 2*CRSR
     ABAJO][ 3*CRSR IZQDA.]";
5Ø15 IF P%=15 THENPRINT
     F$(R1%(Ø));:
     RETURN
5Ø2Ø PRINT F$(R1%(P%+1));
     :RETURN
55ØØ IF P%<2 THENPRINT
     "[CLR/HOME][ 4*CRSR
     ABAJO][ 8*CRSR DCHA.]";
     F$(R1%(14+P%));"[
     2*CRSR ABAJO][ 3*CRSR
     IZQDA.]";:GOTO 551Ø
55Ø5 PRINT "[CLR/HOME][
     4*CRSR ABAJO][ 8*CRSR
```



```
     DCHA.]";F$(R1%(P%-2));
     "[ 2*CRSR ABAJO][
     3*CRSR IZQDA.]";
551Ø IF P%=Ø THENP%=16
5515 P%=P%-1:PRINT
     F$(R1%(P%));"[ 2*CRSR
     ABAJO][ 3*CRSR IZQDA.]";
552Ø IF P%=15 THENPRINT
     F$(R1%(Ø));:
     RETURN
5525 PRINT F$(R1%(P%+1));
     :RETURN
6ØØØ PRINT "[CLR/HOME][
     4*CRSR ABAJO]","[
     5*CRSR DCHA.]";
     F$(R2%(Q%));"[ 2*CRSR
     ABAJO][ 3*CRSR
     IZQDA.]";
6ØØ5 IF Q%=15 THENQ%=
     -1
6Ø1Ø Q%=Q%+1:PRINT
     F$(R2%(Q%));"[ 2*CRSR
     ABAJO][ 3*CRSR
     IZQDA.]";
6Ø15 IF Q%=15 THENPRINT
     F$(R2%(Ø));:
     RETURN
6Ø2Ø PRINT F$(R2%(Q%+1));
     :RETURN
65ØØ IF Q%<2 THENPRINT
     "[CLR/HOME][ 4*CRSR
     ABAJO]","[ 5*CRSR
     DCHA.]";
     F$(R2%(14+Q%));"[
     2*CRSR ABAJO][ 3*CRSR
     IZQDA.]";:GOTO
     651Ø
65Ø5 PRINT "[CLR/HOME][
     4*CRSR ABAJO]","[
     5*CRSR DCHA.]";
     F$(R2%(Q%-2));"[
     2*CRSR ABAJO][ 3*CRSR
     IZQDA.]";
651Ø IF Q%=Ø THENQ%=
     16
6515 Q%=Q%-1:PRINT
     F$(R2%(Q%));"[ 2*CRSR
     ABAJO][ 3*CRSR
     IZQDA.]";
652Ø IF Q%=15 THENPRINT
     F$(R2%(Ø));:
     RETURN
6525 PRINT F$(R2%(Q%+1));
     :RETURN
7ØØØ PRINT "[CLR/HOME][
     4*CRSR ABAJO",,"[
     2*CRSR DCHA.]";
     F$(R3%(R%));"[ 2*CRSR
     ABAJO][ 3*CRSR
     IZQDA.]";
7ØØ5 IF R%=15 THENR%=
     -1
7Ø1Ø R%+R%+1:PRINT
     F$(R3%(R%));"[ 2*CRSR
     ABAJO][ 3*CRSR
     IZQDA.]";
7Ø15 IF R%=15 THENPRINT
     F$(R3%(Ø));:
     RETURN
7Ø2Ø PRINT F$(R3%(R%+1));
     :RETURN
75ØØ IF R%<2 THENPRINT
     "[CLR/HOME][ 4*CRSR
     ABAJO]",,"[ 2*CRSR
     DCHA.]";
     F$(R3%(14+R%));"[
     2*CRSR ABAJO][ 3*CRSR
     IZQDA.]";:GOTO
     751Ø
75Ø5 PRINT "[CLR/HOME][
     4*CRSR ABAJO]",,"[
     2*CRSR DCHA.]";
     F$(R3%(R%-2));"[
     2*CRSR ABAJO][ 3*CRSR
     IZQDA.]";
751Ø IF R%=Ø THENR%=
     16
7515 R%=R%-1:PRINT
     F$(R3%(R%));"[ 2*CRSR
     ABAJO][ 3*CRSR
     IZQDA.]";
752Ø IF R%=15 THENPRINT
     F$(R3%(Ø));:
     RETURN
7525 PRINT F$(R3%(R%+1));
     :RETURN
```

La línea 3000 toma un número al azar, el cual determina si se le ofrecen más avances al jugador, mientras que la línea 3005 activa el *flag* de parada cuando se ha tomado el número apropiado.

Las líneas 3020 a 2065 hacen parpadear las luces de la apuesta (la cantidad apostada) y del stop, y leen lo que ha pulsado el jugador.

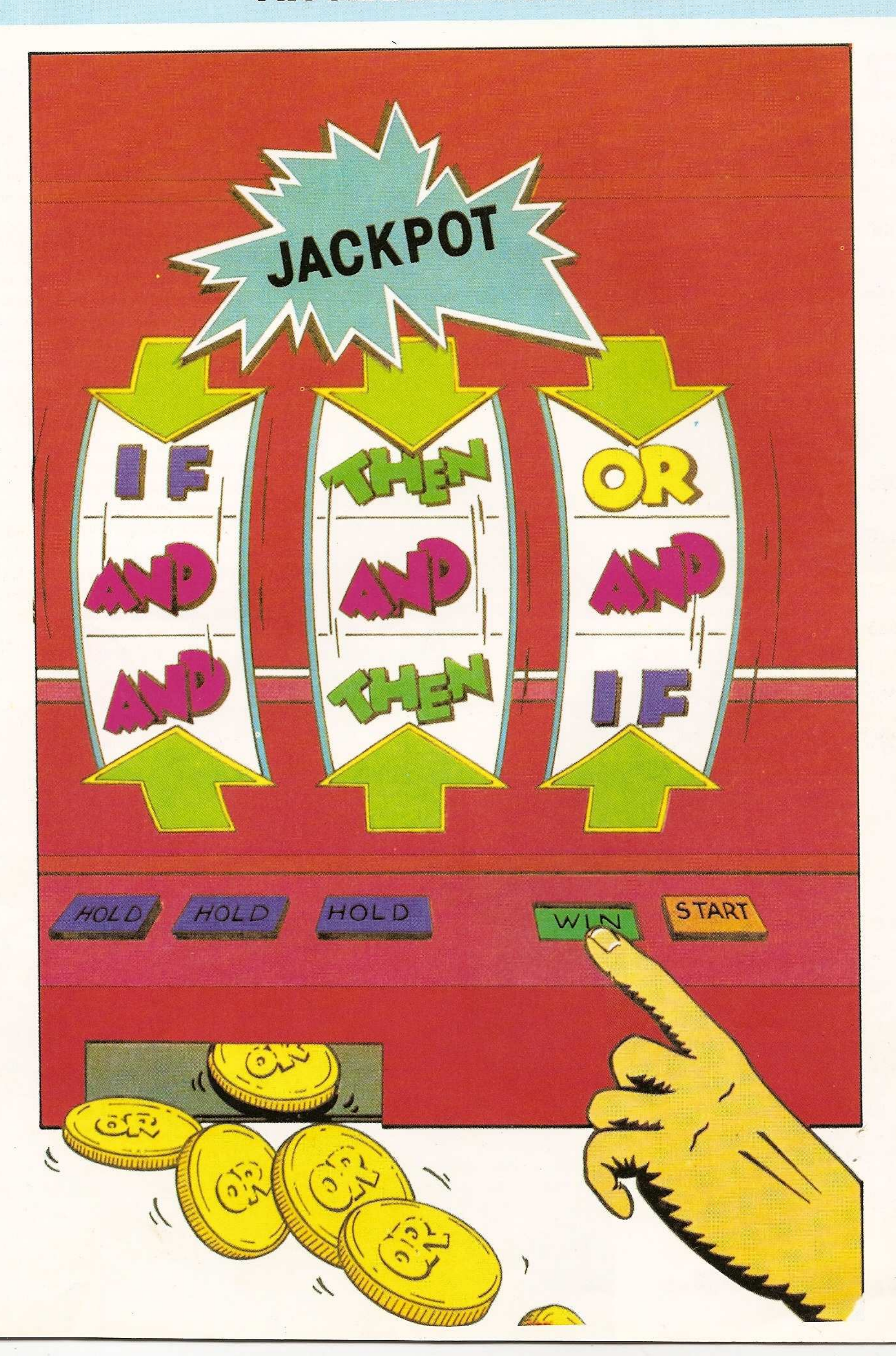
Las líneas 4500 a 4550 constituyen la rutina del avance. Las subrutinas para mover las ruletas son llamadas según la elección o elecciones del jugador. Las líneas 5000 a 5205 mueven hacia arriba la primera ruleta, mientras que las líneas 5500 a 5525 la mueven hacia abajo.

Las líneas 6000 a 6025 y las líneas 6500 a 6525 hacen lo propio con la segunda ruleta, y las líneas 7000 a 7025, así como las líneas 7500 a 7525, controlan la tercera ruleta.

MAS COSAS

```
8ØØØ S=54272:POKE S,
     15Ø:POKE S+1,
     5Ø+DM:POKE S+5,
     Ø:POKE S+6,24Ø:POKE
     S+24,15
8ØØ5 POKE S+4,17:FOR DD=1
     TO DR:NEXT DD:POKE
     S+24,
     Ø:DM=DM+1Ø:RETURN
9ØØØ FOR A=1172 TO 1532
     STEP4Ø:FOR B=A TO
     A+4:POKE B+54272,
     15:NEXT B,A:RETURN
95ØØ PRINT "[CLR/HOME][
     21*CRSR ABAJO]",,"[
     7*CRSR
     DCHA.][COMM+6][SHIFT
     +Z][ 7*CRSR
     IZQDA.][COMM.+1]";
     :IN=INT (M/1ØØ)
95Ø5 RM=M-(IN*1ØØ):PRINT
     MID$(STR$(IN),2);
     "[SHIFT+P]";
     MID$(STR$(RM)+"Ø",2,
     2):RETURN
```

Las líneas 8000 a 8005 son efectos sonoros activados a la parada de las ruletas. Ya tuvimos ocasión de examinarlos en esta revista. Las líneas 9000 a 9505 borran la visualización de la apuesta a la derecha de las ruletas, con lo que aparece apagado y pronto para el siguiente intento en Superfrutas, el juego para apostar a botón pulsado.


JACKPOT
IF
AND
AND
THEN
AND
THEN
OR
AND
IF
HOLD
HOLD
HOLD
WIN
START

EL ZORRO Y LAS OCAS (I)

- CONVIRTIENDO EL ORDENADOR EN UN JUGADOR HABIL
- EL ZORRO, LAS OCAS Y EL AJEDREZ
- TEORIA DEL PROGRAMA

Tu ordenador, para jugar, puede convertirse en un astuto zorro o en unas temerosas ocas.

Observa cómo se pueden aplicar métodos de Inteligencia Artificial a programas escritos capaces de pensar.

En un número anterior de INPUT viste cómo era posible escribir un programa Otelo, en el que el ordenador jugaba de un modo bastante aceptable contra ti. En cualquier caso, con un poco de práctica, seguro que fuiste capaz de vencer al ordenador la mayoría de las veces.

En el curso de las tres partes de programación de JUEGOS que forman este juego, verás cómo también se pueda escribir un programa más sofisticado, a fin de que tu ordenador puede jugar contra ti el juego del Zorro y las Ocas. El programa tiene muchos niveles de habilidad, de modo que puedes hacer que el juego sea tan difícil o tan fácil como quieras.

El juego del Zorro y las Ocas ilustra muchos de los problemas y principios que se incluyen a la hora de escribir uno de los más interesantes y perdurables programas de juegos: el ajedrez. La adquisición de un programa de ajedrez comercial de tipo medio, ejecutable en un microordenador, vale la pena, a no ser que los jugadores posean un nivel muy alto en ajedrez, o conozcan bien las debilidades de los programa de ajedrez.

No se puede escribir un programa de ajedrez que valga la pena en BASIC, pues el ordenador va a tardar un siglo en realizar cada movimiento. En los niveles superiores del Zorro y las Ocas, observarás que el programa tarda mucho más para decidir cada uno de sus movimientos —tal vez más de media hora en los niveles superiores ejecutados en ordenadores más lentos—. El programa aún sería mucho más lento si, en lugar de este juego, intentases escribir un programa de ajedrez.

Necesitamos un juego más simple para mantener alejado nuestro juego-ejercicio del realismo del código máquina. El Zorro y las Ocas nos viene de perlas porque posee muchas de las características del ajedrez, e incluso se juega con el mismo tipo de tablero.

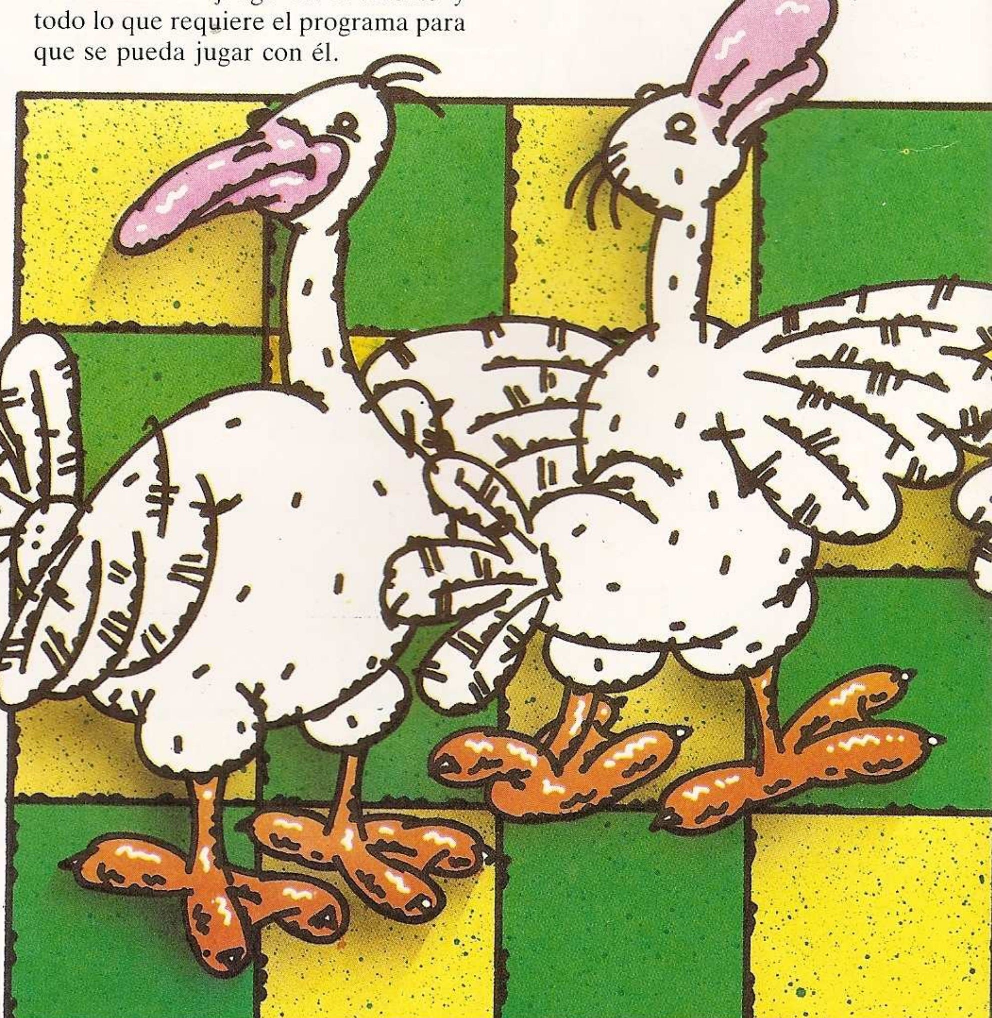
Este artículo se divide en 3 partes.

En esta primera estudiarás los principios que se encierran en un programa de este tipo —y que se irán desarrollando en la segunda y tercera parte del artículo hasta configurar un juego completo—. Pero comencemos observando el juego en sí mismo y todo lo que requiere el programa para que se pueda jugar con él.

EL JUEGO

El juego del Zorro y las Ocas se juega sobre los cuadros blancos de un tablero de ajedrez. Hay un zorro que empieza desde una esquina del tablero y cuatro ocas que comienzan desde el lado opuesto. Un jugador controla al zorro y el otro a las cuatro ocas. El objetivo del juego consiste en lo siguiente: el zorro ha de pasar entre las ocas para llegar al otro lado del tablero, y la misión de las ocas consiste en rodear al zorro.

Con cuatro contra uno, el juego puede parecer un poco injusto, pero resulta que los movimientos de las ocas estan limitados a avanzar únicamente hacia adelante, mientras que el



zorro se puede mover tanto hacia delante como hacia atrás. El programa está escrito de tal modo que el ordenador puede tomar el puesto tanto del zorro como de las ocas, e incluso puede programarse para jugar contra sí mismo.

ABORDANDO LOS PROBLEMAS

El juego del Zorro y las Ocas es un pequeño ajedrez en el sentido de que no existe elemento de suerte en el juego: el resultado depende totalmente de la habilidad de los jugadores. Aunque en teoría, el ordenador podría aprender a jugar a base de probar y equivocarse, como un jugador humano, en el estado actual, éste no es el mejor camino para resolver los problemas que plantea este juego —y además, ¡el programa no cabría en tu microordenador...!



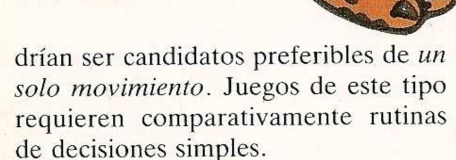
Programar un juego como el del Zorro y las Ocas, o el del ajedrez es realmente un ejercicio de Inteligencia Artificial. Para ofrecer un firme desafío a un oponente humano, el ordenador deberá ser capaz de jugar *inteligentemente*. Desgraciadamente la máquina no puede mirar al tablero y absorber una relación de espacio entre las piezas, como tú haces. En su lugar, la posición en el tablero se convierte en valores numéricos que el ordenador puede entender.

Si quieres escribir un programa para que el ordenador juegue «inteligentemente», deberás tener en cuenta primero la naturaleza del juego. El tipo de juego dictaminará el tipo de programa.

Tal vez exista un *método mejor*, totalmente documentado, que puedas adaptar a tu ordenador. Otros candidatos similares a este tipo de tratamiento serían cosas como: resolver un cubo Rubik, jugar al tres en raya, o las aperturas en ajedrez. En tales casos, tu trabajo se verá bastante simplificado. Si existe un gran elemento de

suerte en el juego, es posible utilizar la estrategia de *un solo movimiento*, en la que el programa examina todas las posibilidades abiertas que diseñase para que elija entre el mejor movimiento.

Los juegos *Ludo* y *Monopoly* podrían ser candidatos preferibles de *un solo movimiento*. Juegos de este tipo requieren comparativamente rutinas de decisiones simples.



En los juegos en los que entran pocos o ningún elemento de azar, como el del Zorro y las Ocas, deberás utilizar una estrategia de *movimiento consecutivo* —observando entre una serie de movimientos—, a fin de investigar las posibles salidas.

El programa encuentra el movimiento más ventajoso desde cada posición y lo realiza.

EVALUACION DE LA POSICION

A fin de permitir al ordenador decidir cuál es el mejor movimiento dentro de una categoría de posibilidades, a cada cuadro del tablero se le asigna un valor numérico. En el Zorro y las Ocas, cuanto más abajo se halle el zorro, tanto mejor para él, pues recuerda que éste gana cuando llega al otro extremo del tablero.

Las filas de casillas están numeradas alternativamente de izquierda a derecha y luego de derecha a izquierda. Ello asegura que las ocas tiendan a permanecer en una línea recta —su configuración más fuerte— y el cuadro que llevará asignando el valor más alto se va alternando entre el final izquierdo y derecho de la fila.

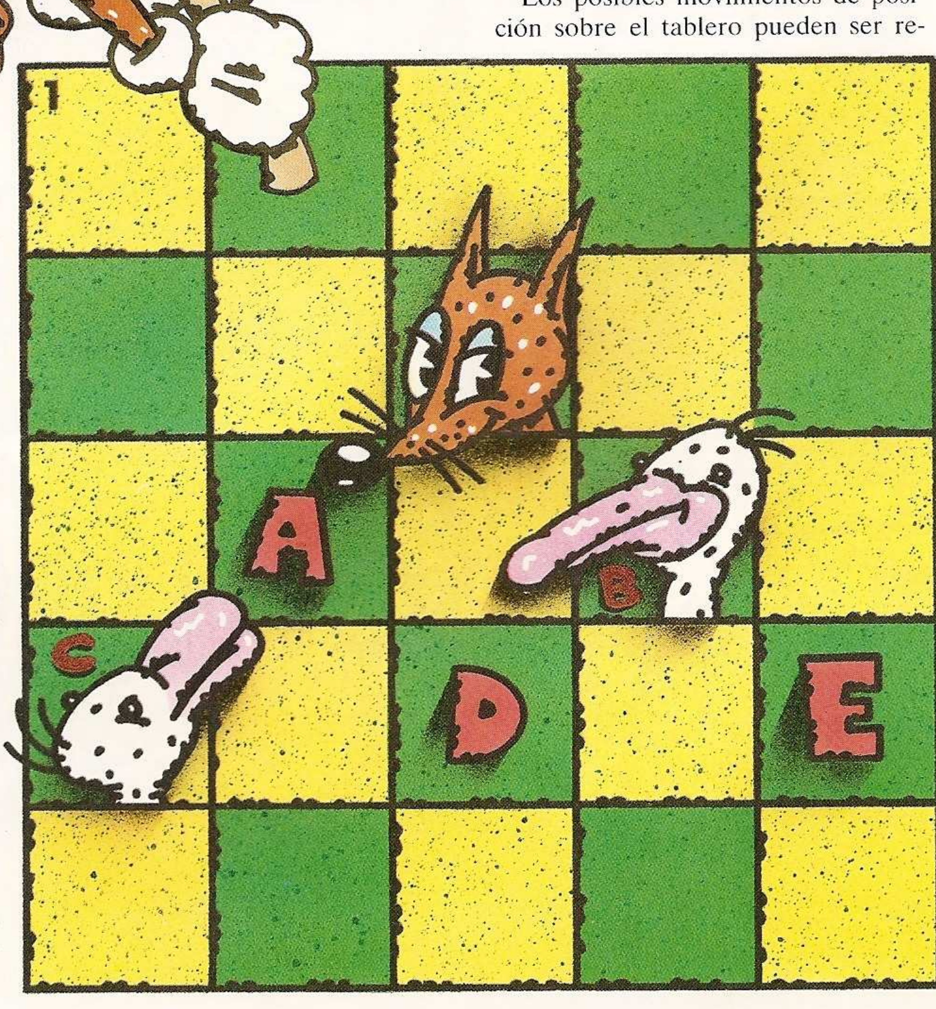
Además de la sencilla evaluación posicional, los cinco cuadros blancos delante del zorro tienen un significado especial en el juego. Si observas la figura 1, verás cinco cuadrados con las letras de A a E.

Si sólo hay una oca situada en estos cuadros, o si hay dos ocas, y no están en las posiciones A o B, gana el zorro; o bien, si hay tres y éstas están en las posiciones ACD, BDE, ACE o BCE, el zorro aún sigue ganando.

Al comienzo de cada turno de juego, cuando el ordenador juegue, o bien a ser zorro o las ocas, el programa salta a una subrutina que revisa la posición (o configuración) de todas las piezas y las convierte en un solo número que utiliza el programa para decidir qué movimiento realizar. Una vez tomada la decisión, el número vuelve a convertirse en una configuración.

ARBOL DE INVESTIGACION

Los posibles movimientos de posición sobre el tablero pueden ser re-



presentados con una escritura en forma de árbol, cuyas ramas emanan de la posición de las piezas (raíces). Por ejemplo, desde la posición en la figura 2, se podría trazar el primer nivel del árbol en la figura 3. Si miras dos movimientos más adelante el árbol se vuelve más complejo que en el segundo nivel —ten en cuenta que no siempre hay razón para que partan cuatro ramas de cada cuadro, ya que la pieza podría estar bloqueada por otra, o podría hallarse en el extremo del tablero.

3

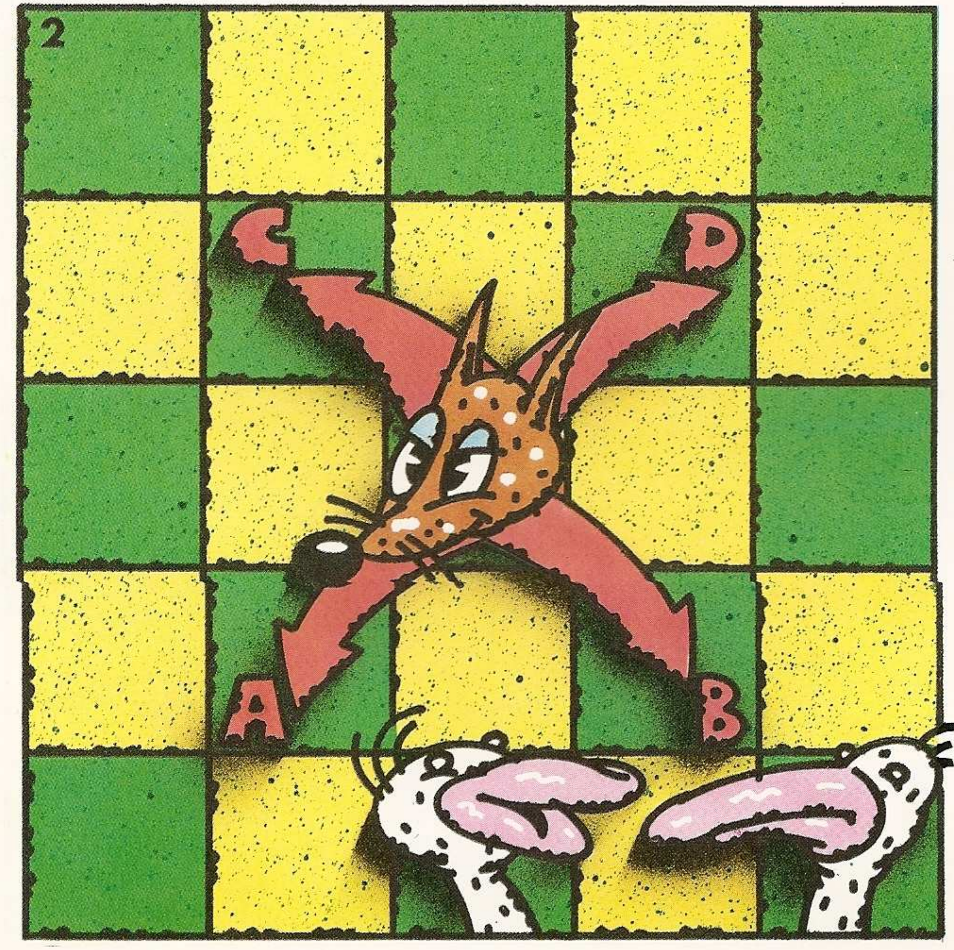
A B C D

BEST

ACELERANDO EL PROGRAMA

Con el número de cálculos a ejecutar, el BASIC puede resultar muy lento, pero existen tres caminos para acelerar el programa. En primer lugar, puedes asegurarte de que el programa no molesta para evaluar el próximo movimiento del adversario, en caso de que tu movimiento haya ganado la partida. En cualquier caso, ello sólo ahorra tiempo al final, pero no durante la partida.

En segundo lugar, se puede llegar muy frecuentemente a la misma configuración desde muy diversas vías. Si éste es el caso, valdrá la pena el esfuerzo de construir una tabla de configuraciones comunes y de valores calculados asociados con ellos. Una tabla de este tipo detiene el programa volviendo a asignar cada vez la posición individualmente, pero no hay duda en la conveniencia de grabar una configuración que requiriese menos tiempo en ser evaluada que lo que se tardaría en encontrarla en la tabla. En teoría, la tabla sólo puede ser utilizada cuando el programa ha avanzado tres *pliegues* (turnos completos de cada jugador) o más. La práctica del juego demuestra que no hay razón de registrar nada en la tabla hasta que se hayan avanzado, como mínimo, cinco *pliegues*.



En tercer lugar, se puede utilizar el denominado algoritmo alfa-beta. Este fue descubierto a principios de los años sesenta por investigadores del campo de Inteligencia Artificial, y se emplea cada vez que el registro del árbol necesita examinar más de un *pliegue* de un árbol.

Si observas la figura 3 podrás ver la evaluación de los posibles movimientos de una posición del zorro. El programa evaluará completamente la rama A y luego pasará a la rama B. Si en cualquiera de las fases de evaluación de B se detecta un resultado erróneo, se rechazarán todos los movimientos en B. El mejor movimiento efectuado hasta el momento queda registrado en la memoria del programa y comparado con los resultados en cada rama en turno. Si se encuentra en un punto cualquiera un resultado erróneo, ello provocará que toda la rama sea rechazada.

El algoritmo alfa-beta, adquiere mayor importancia realmente a medida que aumenta la complejidad del árbol. Si el árbol aumenta considerablemente de tamaño, puede permi-

tirse descartar alrededor del 99,8 % de posibilidades a un nivel primario, con un ahorro similar de tiempo. Pero en este juego el árbol no se aproximará a este nivel de complejidad.



EJECUTANDO EL PROGRAMA

Ahora que ya tienes algo de idea acerca de la teoría que alberga la escritura del programa de un juego de estas características, puedes pasar a la primera sección del programa. Consiste en trazar los gráficos, aunque no verás nada en este nivel. Si ejecutas el programa con RUN, no olvides grabarlo (SAVE) en un cassette que utilizarás en la próxima parte del programa.

Antes de utilizar el programa Commodore deberás llevar hacia arriba el inicio del BASIC a fin de dejar espacio libre para el juego de gráficos. Escribe:

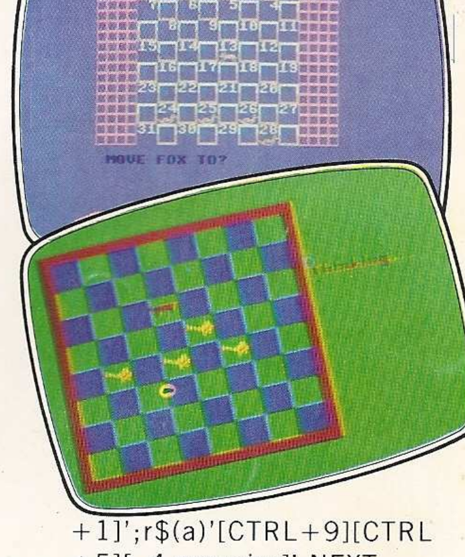
```
POKE 44,18
POKE 256*18,0
NEW
```

cada vez antes de cargar el programa.

```
1 IF PEEK(44)=8 THENPOKE
  44,18:POKE 256 * 18,
  Ø:PRINT 'carga de nuevo el
  programa':NEW
2 POKE 5328Ø,6:POKE 53281,
  6PRINT '[SHIFT+CLR/
  HOME][CTRL+8]definiendo
  gráficos...';GOSUB 4ØØØ
4ØØØ POKE 56334;Ø:POKE 1,35
4Ø1Ø FOR z=Ø TO 1Ø23:POKE
     2Ø48 + z,PEEK(53248 +
     z):NEXT z
4Ø2Ø POKE 1,39:POKE 56334,1
4Ø3Ø FOR z=68Ø TO 711:READ
     x:POKE 2Ø48 + z,x:NEXT z
4Ø4Ø FOR z=3329 TO
     3334:POKE z,129:NEXT
     z:POKE 3328,255:POKE
     335,255
4Ø5Ø RETURN
5ØØØ DATA Ø,Ø,Ø,7,2Ø5,123,6
     Ø,15,12,2Ø,31,152,248,
     216,48,224
5Ø1Ø DATA 2Ø,28,55,127,15,
     2Ø,4Ø,72,Ø,Ø,248,252,2
     5Ø,4Ø,2Ø,2Ø
```

DIBUJANDO EL TABLERO

```
31Ø f=FN a(ABS(p)) − 31:b=p /
    b(f):IF b<Ø THENb=b +
    bx:f=31 − f
32Ø c=b / b(28):FOR a=7 TO Ø
    STEP − 1:r$(a)=b$(INT (c),
    1 AND a):c=(c − INT (c)) *
    16:NEXT a
33Ø r$(f / 4)=LEFT$(r$(f / 4),FN
    c(f)) + f$(f / 4 AND 1) +
    RIGHT$(r$(f / 4),1 2 − FN
    r(f))
34Ø PRINT '[SHIFT+CLR/
    HOME]';:FOR a=Ø TO
    7:PRINT TAB(7);
    '[CTRL+9][CTRL+5-[ 4
    espacios][CTRL+Ø][CTRL+
    4]';s$(a)'[CTRL+9][CTRL+
    5][4espacios]'
35Ø PRINT TAB(7);
    '[CTRL+9[CTRL+5][ 4
    espacios][CTRL+Ø][COMM.
```




```
    +1]';r$(a)'[CTRL+9][CTRL
    +5][ 4 espacios]':NEXT
    a:PRINT
    '[CTRL+1]':RETURN
```

Las líneas 310 a 350 muestran el tablero con las cinco piezas en posición. La subrutina es llamada una vez a cada turno del zorro y las ocas.


Chip
¡DEBERÍA DARTE VERGÜENZA!
SI NO TIENE...
¡COMO SE ENTERE EL CPU TE VAN A CAER LAS PATILLAS...!
¡UY, SI SE ENTERA...!
¿PERO TÚ QUE TE CREES QUE ES ESTO?!
¿LA FERIA DE SEVILLA?!
SÓLO LE FALTAN LAS CASTAÑUELAS
YA OS HE DICHO MIL VECES QUE YO NO TENGO LA CULPA...
¡BAH! NO NOS VENGAS OTRA VEZ CON LA TONTERÍA ESA DEL SIMULADOR DE VUELO...
¡QUE SÍ, HOMBRE! LO QUE PASA ES QUE A VOSOTROS APENAS OS HA AFECTADO... YA VERÉIS LA PRÓXIMA VEZ QUE...
¡ESCUCHAD! PARECE QUE VA A JUGAR DE NUEVO...
¡AJÁ! YA HA EMPEZADO... ¡EL AVIÓN ESTÁ DESPEGANDO...!
BRRRRRRRRR
RRRRRRRRRRR
AHORA SE VA ACERCANDO A UN ARCO IRIS...
... Y DENTRO DE POCO LO ATRAVESERÁ...
¡AHORA! ¿NO LO NOTÁIS EN LOS CIRCUITOS?
S... S... SÍ...
¿QUÉ? ¿OS CONVENCÉIS AHORA? YA OS DIJE QUE ESE ARCO IRIS DESTEÑÍA...
¿Y... Y ESO NO SE VA?
OYE... PUES NO QUEDA MAL...
¡JO!
SIRVENT. 87.

LA LUNA A TUS PIES

En este formidable juego vas a necesitar de toda la habilidad y sangre fría de que seas capaz para maniobrar el módulo lunar de forma que pueda alunizar perfectamente.

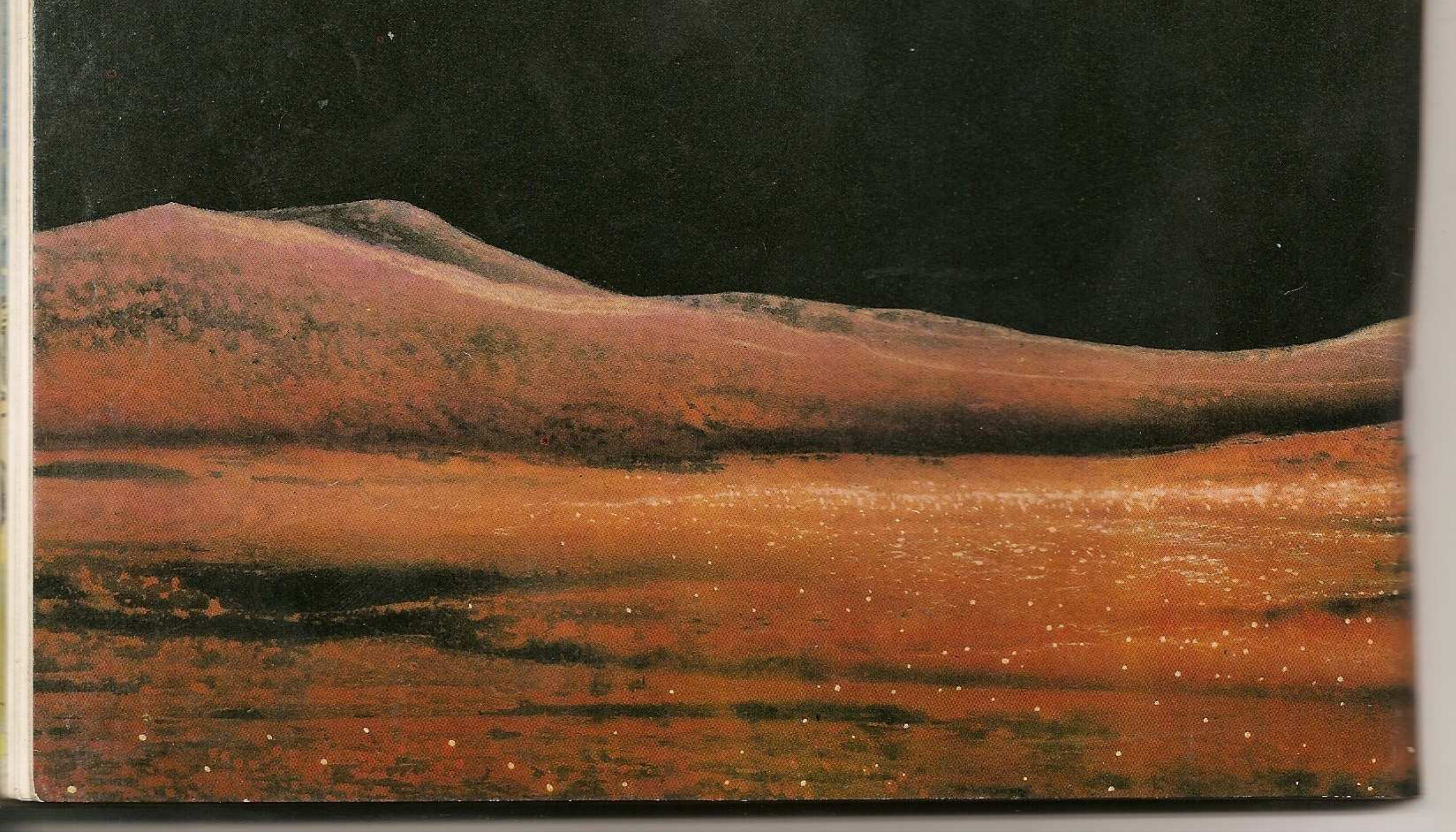
La programación de juegos no tiene por qué generar complicados programas para producir un juego independiente y completo. Aquí te presentamos una versión del célebre programa *Módulo lunar* (*Lunar lander*) que contiene gráficos en alta resolución y un control total sobre el aparato.

El juego es completo y muy variado según lo presentamos, pero tú tienes la posibilidad de adaptarlo a tus preferencias personales. Por ejemplo, puede que te guste añadir una rutina del tipo «¿otro intento?» para evitar volver otra vez al RUN una vez que se ha concluido el descenso. O bien puede que desees alterar los gráficos y los sonidos. Todo depende de ti.

CONTROLES

Las teclas de cursor derecha y cursor izquierda, así como la tecla de Commodore para alunizar.

```
5 REM ------- INPUT
  COMMODORE -------
6 REM
1Ø HIRES1,Ø:MULTI7,4,
  4:COLOUR7,Ø:POKE54296,
  15:POKE54277,190:POKE
  54278,248
15 POKE 54278,248
7Ø FOR Z=2Ø TO 16Ø STEP
  2Ø:Y=
  179-RND(1)*4Ø:LINE
  Z-2Ø,179,Z-1Ø,Y,3
72 FOR ZZ=1 TO 3:PLOT RND
  (1)*16Ø,RND(1)*150,ZZ:
  NEXTZZ
74 LINEZ-1Ø,Y,Z,179,3:NEXT
  Z:PAINTØ,199,1:TEXT64,
  192,"[SHIFT+N][SHIFT+L]
  [SHIFT+@][SHIFT+M]",2,
  1,8
76 LINE Ø,199,159,199,2
11Ø LX=RND(1)*248:LY=15+
  RND(1)*1Ø:XV=RND(1)*15
  -8:YY=Ø:F=246
115 GOSUB1Ø4Ø
12Ø GOSUB1Ø1Ø:GOSUB2ØØØ:
  MULTI7,RND(1)*2+4,RND
  (1)*8+1
13Ø IF LY<192 THEN 12Ø
14Ø IF LX<72 OR LX>88 OR
  YV>4 THEN 16Ø
15Ø PRINT "[SHIFT+CLR/
  HOME][CTRL+5][CRSR
  ABAJO][CRSRDCHA.][CTRL
  +9]FELICIDADES,HAS
  CONSEGUIDOATERRIZAR!"
155 POKE 54276,33: FOR Z=1
  TO 255:POKE 54273,
  A:POKE 54273,255-Z:
  NEXT Z: GO TO 17Ø
16Ø PRINT "[SHIFT+CLR/
  HOME]:CENTRE"[CTRL+5]
```



- UN JUEGO COMPLETO
- HABILIDAD Y DECISION
- GRAFICOS LUNARES
- VELOCIMETRO
- CONTROL DE ATERRIZAJE
- ADAPTAR EL PROGRAMA
- EFECTOS SONOROS
- DESASTRES
- PROGRAMAS CON EXITO
- EL MODULO LUNAR

```
     CATASTROFICO
     ATERRIZAJE":POKE54276,
     129
163  FORZ=1TO1ØØ:COLOUR7,
     RND(1)*2:POKE54273,Z
165  FOR ZZ=1 TO 1Ø: NEXT ZZ,
     Z
17Ø  POKE 54276,Ø:NRM: POKE
     198,Ø:END
1Ø1Ø LX=LX+XV:LY=LY+YV:
     POKE54276,17:S=255-
     (255ANDLY):POKE54273,
     S
1Ø2Ø IF LY<13 THEN RETURN
1Ø3Ø IF LX<Ø THEN LX=151
1Ø35 IF LX>151 THEN LX=Ø
1Ø4Ø TEXT LX,LY,"[SHIFT+X]",
     4,1,8:GOSUB3ØØØ
1Ø5Ø TEXTLX,LY,"[SHIFT+X]",
     4,1,8:POKE54276,16:
     RETURN
2ØØØ YV=YV+.5:IF LY<13
     THEN RETURN
2Ø1Ø GETK$:IFPEEK(653)=2
     ANDF>3THENYV=YV-1:F
     =F-3:RETURN
2Ø2Ø IF K$ = "[CRSR ABAJO]"
     THEN XV = XV - .5: F = F
     - 1: RETURN
2Ø3Ø IFK$="[CRSRDCHA.]"
     THENXV=XV+.5:F=F-1
2Ø4Ø RETURN
3ØØØ TEXT 1,1,"GAS
     :"+STR$(F),1,1,8
3Ø1Ø V=2*YV:IFABS(V)>122
     THENV=122*SGN(V)
3Ø2Ø TEXT75,1,"VELD:"+STR$
     (V),1,1,8
3Ø21 TEXT41,1,STR$(F),Ø,1,8
3Ø22 TEXT123,1,STR$(V),Ø,1,8
3Ø3Ø RETURN
```



EL ORDENADOR A LA ESCUCHA

Hasta ahora, el único medio de que disponíamos para comunicarnos con el ordenador era el teclado o el *joystick*, pero utilizando el conversor analógico-digital del que dispone es posible que también nos obedezca a distancia.

En concreto utilizaremos la voz para comandarlo, ya que es la forma más natural de comunicación entre nosotros. Por este motivo los medios de telecomunicación habituales están basados en la palabra; de esta forma, dos ordenadores que deseen comunicarse a distancia deberán hacerlo «hablando».

El problema principal es, por tanto, conseguir que el ordenador «oiga». De esto se encargará un económico dispositivo electrónico que actuará como *vox controller*, es decir, controlador de voz. La única misión de este circuito será la de proporcionar una señal al ordenador cuando mediante un micrófono se capte un sonido de intensidad adecuada. Aunque parezca algo simple, las aplicaciones que de él se derivan son numerosas. En el artículo se describen algunas interesantes, aunque tú puedes encontrar otras muchas.

EL CIRCUITO

En la figura 1 se representa el esquema electrónico del controlador de voz, de muy fácil construcción. Puede verse que sólo utiliza dos componentes activos con lo que resulta barato y fiable.

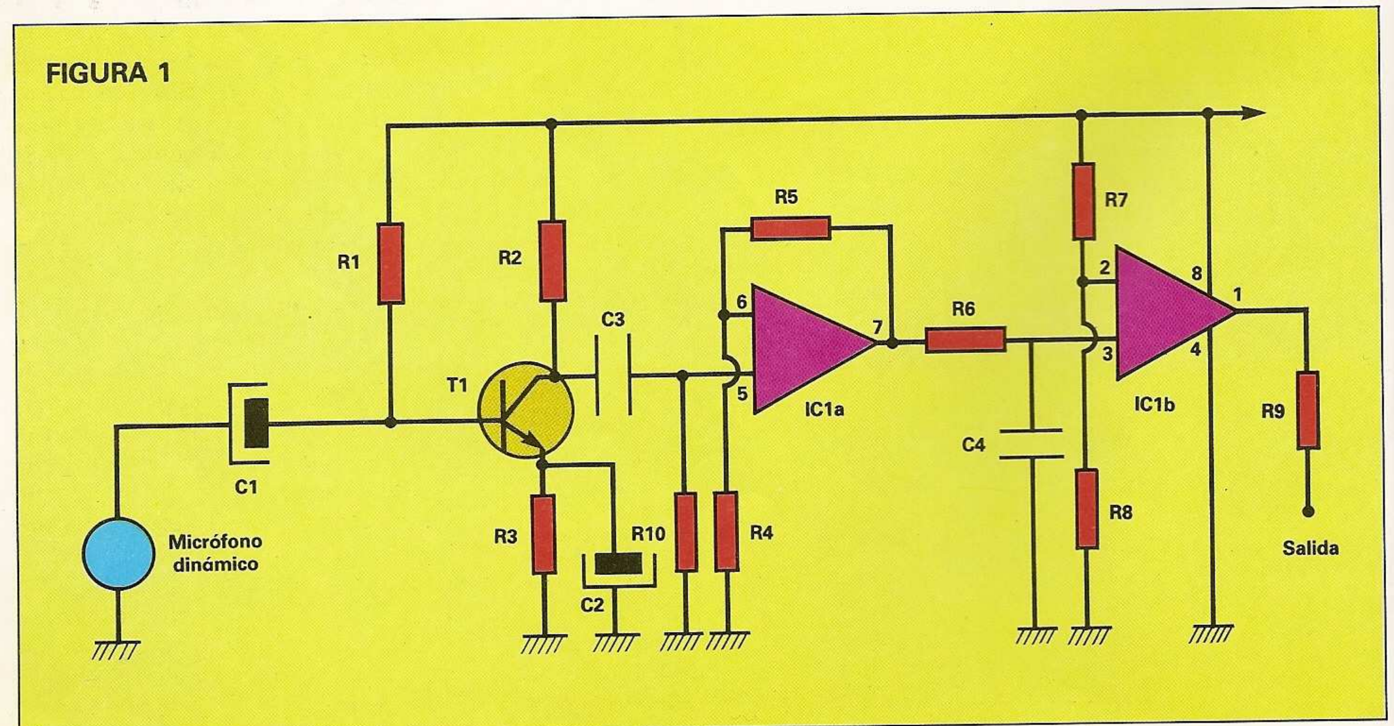
El circuito consta de tres etapas. La primera de ellas está formada por un transistor y su red de polarización (R1, R2 y R3). Su misión es amplificar la débil señal que proporciona el micrófono dinámico. La elección del micrófono no es crítica, pudiendo incluso utilizarse un altavoz convencional en lugar del mismo.

La segunda etapa es un amplificador de elevada ganancia formado por el primer operacional del LM 358, en configuración no inversora. A la salida del mismo se obtiene una señal comprendida entre 0 y 5 V que se aplica a un filtro paso-bajo, formado por R6 y C4, cuya función es eliminar las altas frecuencias.

La última etapa es un comparador que proporciona 5 V a la salida cuando la tensión en la patilla 3 supera un cierto umbral fijado por R7 y R8. Se puede ajustar este umbral haciendo el dispositivo más sensible bajando el valor de R8, aunque no excesivamente, ya que cualquier ruido lo dispararía.

Los valores propuestos son los más adecuados para que un débil sonido captado por el micrófono active la salida, sin que el ruido ambiental lo afecte.

En la figura 2 se representan las conexiones del transistor, circuito integrado y conector al ordenador.



USO COMO MODEM TELEFONICO

Quizá sea ésta la aplicación más interesante que hemos encontrado del circuito. Todos sabemos que la información que maneja el ordenador es binaria, es decir, señal («1») o no señal («0»). Por lo tanto sólo es necesario distinguir entre dos niveles, que es precisamente lo que hace el controlador de voz. Como la sensibilidad del aparato es elevada, podemos utilizar la línea telefónica, que proporciona poca intensidad sonora, para establecer la comunicación. Podremos transmitir información de un ordenador a otro a través del teléfono.

Es evidente que un ordenador debe funcionar como transmisor y el otro como receptor. Con el objeto de economizar, y sin que por ello la calidad se vea gravemente afectada, emplearemos el altavoz del televisor como fuente emisora; el propio ordenador se encargará, con el chip de sonido, de generar la señal sonora mediante un programa adecuado. Es en el receptor donde se utilizará el controlador, acoplado al teléfono.

El programa desarrollado permitirá transmitir caracteres en formato ASCII, ya que de esta forma se pueden aprovechar una serie de subrutinas del sistema operativo de los ordenadores y los programas quedan simplificados. Para que la transmisión sea eficiente se precisan dos bits adicionales a los siete del código ASCII, que son el bit de inicio y el de parada. El primero se representa por un nivel lógico 1 e indica al receptor que a continuación se transmite un carácter. El bit de parada se representa por un 0 y se envía al final de cada carácter para restablecer un nivel bajo en la línea. Esto es conocido técnicamente como transmisión en serie asíncrona, ya que los datos se transmiten bit tras bit y en cualquier momento.

En la figura 3 se representa cómo queda el tren binario al enviar el carácter «A». Puede apreciarse que el primer bit que se envia tras el bit de inicio es el correspondiente al bit de menos peso del código de la «A» (65).

El problema con el cual nos encontramos es que el tren de bits no se puede transmitir tal cual por el teléfono. Lo que debe hacerse es asignar una forma de onda a cada bit; lo más sencillo es hacer corresponder un sonido al valor «1», y ningún sonido al valor «0». Esto es lo que se conoce en el lenguaje de las telecomunicaciones con el nombre de modulación por corrimiento de amplitud o ASK. El sonido que se le asigne al «1» debe ser el adecuado para que se propague bien por el teléfono, preparado sólo para transmitir la voz. Por este motivo se elige un tono entre 1 y 2 KHz para la emisión.

Debido a que se trata de un modem experimental en el cual los costes se han reducido al mínimo, las velocidades de transmisión no pueden ser demasiado elevadas. Las malas características del acoplamiento acústico limitan la velocidad a 25 Bauds para una tasa de error no demasiado alta, es decir, se pueden transmitir tres caracteres cada segundo. Se podría aumentar si no importa una mayor frecuencia de errores. La aplicación básica a la que puede ir destinado es la transmisión de mensajes entre ordenadores. También es posible transmitir caracteres gráficos para formar dibujos, e incluso gráficos en alta resolución, con algunas modificaciones en los programas. También es posible enviar todos los caracteres de control (Return, Del, etc.), ya que tienen su correspondiente código ASCII.

Los ordenadores deben estar controlados por dos programas distintos, uno encargado del emisor y otro del receptor. El programa del emisor tiene dos partes: una en BASIC y otra en código máquina. El programa en BASIC se encarga de pedir el mensaje a enviar (máximo 256 caracteres), convertirlo a código ASCII, guardarlo en memoria y poner en marcha el programa emisor propiamente dicho. Si se desea enviar mensajes más largos sólo es necesario ejecutar varias veces este programa o modificarlo según las necesidades.

El programa en código máquina puede ser cargado con la ayuda de un ensamblador o bien utilizando la versión BASIC. Funciona del siguiente modo: en primer lugar se ajustan los valores del chip de sonido para que emita un tono de unos 2 KHZ. A continuación se ajustan los contadores internos para controlar de forma precisa la duración de cada pulso o bit. Esta debe ser de 40 milisegundos. Al transcurrir este intervalo de tiempo se genera una interrupción no enmascarable, que hace saltar el programa a una

Lista de componentes:
R1 = 100 Kohm
R2 = 150 Ohm
R3 = 270 Ohm
R4 = 390 Ohm
R5 = 68 Kohm
R6 = 10 Kohm
R7 = 18 Kohm
R8 = 3,9 Kohm
R9 = 12 Kohm
R10 = 6,8 Kohm
C1 = 470 µF
C2 = 470 µF
C3 = 100 nF
C4 = 27 nF
T1 = BC 337 o similar
IC1 = LM 358
Conexiones al ordenador:
+5 Voltios a Pin 7 del port 1 de juegos
Masa a Pin 8
Salida a Pin 9

subrutina de servicio de interrupción. Por eso se hace necesario modificar los punteros de la NMI de forma que apunten a una rutina propia ubicada en C0A0. Tras hacer esto comienza la emisión propiamente dicha. En primer lugar se envía el bit de inicio para lo cual se pone el volumen al máximo durante 40 ms. A continuación se carga el acumulador con el carácter a transmitir y se va rotando para extraer los bits uno a uno. Si el bit extraído es un «1» se pone el volumen al máximo y si es un «0» se desactiva. El altavoz del televisor reproduce estas variaciones de forma más o menos correcta. El proceso se itera tantas veces como es necesario para transmitir la totalidad del mensaje. Al acabar se ajustan las interrupciones a los valores iniciales y se devuelve el control al sistema operativo.

LISTADO EN ENSAMBLADOR DEL PROGRAMA EMISOR (C 64)

```
CØØØ: LDA # $45
CØØ2: STA $D4Ø1
CØØ5: LDA # $FØ
CØØ7: STA $D4Ø6
CØØA: LDA # $ØØ
CØØC: STA $D4Ø5
CØØF: STA $D4Ø2
CØ12: LDA # $Ø4
CØ14: STA $D4Ø3
CØ17: LDA # $41
CØ19: STA $D4Ø4
CØ1C: LDA # $4E
CØ1E: STA $DDØ4
CØ21: STA $DDØ5
CØ24: LDA # $Ø1
CØ26: STA $DDØ6
CØ29: LDA # $ØØ
CØ2B: STA $DDØ7
CØ2E: LDA # $11
CØ3Ø: STA $DDØE
CØ33: LDA # $51
CØ35: STA $DDØF
CØ38: LDA # $AØ
CØ3A: STA $Ø318
CØ3D: LDA # $CØ
CØ3F: STA $Ø319
CØ42: LDA $DDØD
CØ45: LDA # $82
CØ47: STA $DDØD
CØ4A: LDX # $ØØ
CØ4C: TXA
CØ4D: PHA
CØ4E: LDA $C3ØØ,X
CØ51: PHA
CØ52: LDY # $FF
CØ54: CPY # $ØØ
CØ56: BNE $CØ54
CØ58: LDA # $ØF
CØ5A: STA $D418
CØ5D: LDY # $FF
CØ5F: CPY # $ØØ
CØ61: BNE $CØ5F
CØ63: LDA # $ØØ
CØ65: STA $D418
CØ68: LDX # $Ø7
CØ6A: LDA # $ØØ
CØ6C: STA $D418
CØ6F: PLA
CØ7Ø: ROR
CØ71: PHA
CØ72: BCC $CØ79
CØ74: LDA # $ØF
CØ76: STA $D418
CØ79: LDY # $FF
CØ7B: CPY # $ØØ
CØ7D: BNE $CØ7B
CØ7F: DEX
CØ8Ø: BNE $CØ6A
CØ82: LDA # $ØØ
CØ84: STA $D418
CØ87: LDY # $FF
CØ89: CPY # $ØØ
CØ8B: BNE $CØ89
CØ8D: PLA
CØ8E: PLA
CØ8F: TAX
CØ9Ø: INX
CØ91: CPX # $ØØ
CØ93: BNE $CØ4C
CØ95: JSR $FDA3
CØ98: RTS
CØAØ: LDA $DDØD
CØA3: AND # $Ø2
CØA5: BNE $CØAA
CØA7: JMP $FE56
CØAA: INY
CØAB: RTI
```

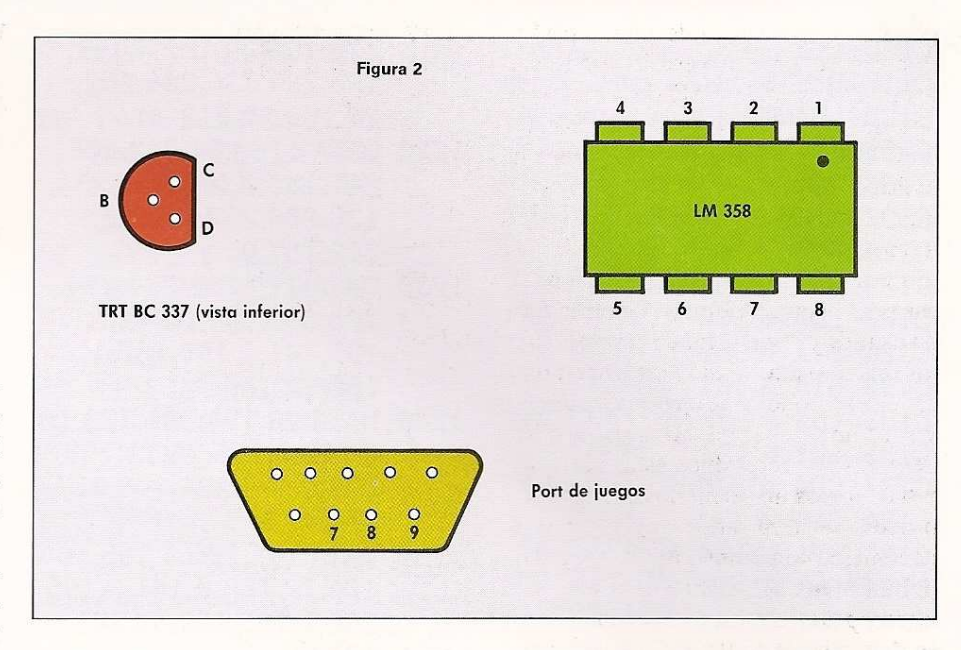
La versión en BASIC es la siguiente:

```
1ØØØ FOR I = 49152 TO 493Ø4:
     READ A: POKE I,A: NEXT
1Ø1Ø DATA 169,69,141,1,212,
     169,24Ø,141,6,212,169,
     Ø,141,5,212,141,2,212,
     169,4
1Ø2Ø DATA 141,3,212,169,65,
     141,4,212,169,78,141,4,
     221,141,5,221,169,1,
     141,6
1Ø3Ø DATA 221,169,Ø,141,7,
     221,169,17,141,14,221,
     169,81,141,15,221,169,
     16Ø,141
1Ø4Ø DATA 24,3,169,192,141,
     25,3,173,13,221,169,
     13Ø,141,13,221,162,Ø,
     138,72,189
1Ø5Ø DATA Ø,195,72,16Ø,255,
     192,Ø,2Ø8,252,169,15,
     141,24,212,16Ø,255,
     192,Ø,2Ø8
1Ø6Ø DATA 252,169,Ø,141,24,
     212,162,7,169,Ø,141,24,
     212,1Ø4,1Ø6,72,144,5,
     169,15
1Ø7Ø DATA 141,24,212,16Ø,
     255,192,Ø,2Ø8,252,2Ø2,
     2Ø8,232,169,Ø,141,24,
     212,16Ø
1Ø8Ø DATA 255,192,Ø,2Ø8,252,
     1Ø4,1Ø4,17Ø,232,224,Ø,
     2Ø8,183,32,163,253,96
1Ø9Ø FOR I = 49312 TO 49323:
     READ A: POKE I,A: NEXT
11ØØ DATA 173,13,221,41,2,
     2Ø8,3,76,86,254,2ØØ,64
2ØØØ PRINT "MENSAJE"?: FOR I
     = 1 TO 1ØØØ: NEXT: L =
     Ø: PRINT"CLR"
2Ø1Ø GET A$: IF A$=""THEN
     2Ø1Ø
2Ø2Ø L = L + 1
2Ø3Ø IFA$ = CHR$ (2Ø)
     THENM$= LEFT$ (M$,L-
     2): L = L-2: PRINT"
     (CLR)": A$ = ""
2Ø4Ø IFA$= "(F1)"THEN21ØØ
2Ø5Ø M$=M$+A$:
     PRINT"(HOME)"M$
2Ø6Ø IFL=255THEN PRINT
     "ENVIO MENSAJE
```

```
        ":GOTO21ØØ
2Ø7Ø GOTO2Ø1Ø
21ØØ POKE 49298,L–1
211Ø  FOR I=1 TO L–1:
      POKE4992Ø+I–1,
      ASC(MID$[M$,I,1]):NEXT
212Ø POKE 49920+L–1,Ø
213Ø SYS 49152
214Ø END
```

Para aquellos que utilicen ensamblador sólo es necesario copiar a partir de la línea 2000. El programa pide el mensaje y lo va mostrando en pantalla. Puede corregirse mediante la tecla DEL y para acabar debe oprimirse F1. Si el mensaje contiene más de 256 caracteres comienza la emisión automáticamente y será necesario hacer RUN2000 para continuar la emisión del resto del mensaje.

El programa receptor está descrito en su totalidad en código máquina y para su ejecución basta escribir SYS 49408, una vez almacenado en memoria mediante un ensamblador o el cargador BASIC. Tanto el programa emisor como el receptor pueden coexistir en memoria. De esta forma es posible establecer una comunicación en los dos sentidos. El programa receptor funciona del siguiente modo: inicialmente espera que el valor de la posición 54297, que corresponde al conversor A/D, sea inferior a 129, lo que indica que se ha captado alguna señal. Para evitar que un posible ruido indeseado sea tomado como el bit de inicio se toman cuatro muestras más en instantes posteriores. Si en todas ellas se ha detectado señal se considera que el bit de inicio ha sido recibido; en caso contrario se vuelve al bucle de espera. Una vez detectado éste se comienza a tomar muestras del tren de bits y se guardan en una tabla. Por cada bit se toman 32 muestras, por lo que entre muestra y muestra se deberá esperar 1,25 ms. De esto se encarga el bucle ubicado entre C11A y C11E. Aquí no es necesario usar el timer porque al tratarse de transmisión asíncrona, el bit de inicio desencadena todo el proceso. Una vez almacenados en la tabla todos los valores correspondientes a un carácter se procede a su identificación. Para ello se evalúa el valor más frecuente de las 32 muestras de cada bit, protegiéndose así el sistema contra posibles errores. Una vez identificado el código ASCII se llama a la subrutina E716 que lo imprime en pantalla. Si el carácter detectado ha sido el 00, se interpreta que la transmisión ha concluido y se devuelve el control al sistema operativo. El carácter 00 ya lo introduce automáticamente el programa emisor.



Para ejecutar una transmisión de datos debe procederse del siguiente modo:

—Cargar los programas adecuados en cada ordenador.
—Enchufar el controlador de voz al receptor.
—Situar el micrófono del teléfono junto al altavoz del televisor seleccionando un volumen medio para emitir. El receptor deberá situar el auricular del teléfono junto al micrófono del controlador.
—Emitir un mensaje de prueba preestablecido, por ejemplo enviando reiteradamente la misma letra.
—Ajustar el volumen del televisor y la distancia hasta el teléfono hasta conseguir una buena transmisión.
—Emitir el mensaje deseado.

Con un buen ajuste se han conseguido tasas de error del orden del 1 %. Si esta tasa es considerada demasiado alta, descendiendo la velocidad de transmisión, puede rebajarse ostensiblemente. Se comprueba que el sistema es bastante insensible a los ruidos propios del canal telefónico.

La velocidad de transmisión se establece en las siguientes posiciones de memoria:

—para el emisor: en C01D, C025 y C02A.
—para el receptor: en C10A y C11B.

Debe tenerse presente, que en cualquier caso, las velocidades de emisión y recepción deben ser iguales.

LISTADO EN ENSAMBLADOR DEL PROGRAMA RECEPTOR (C 64)

```
C1ØØ: LDA $D419
C1Ø3: CMP # $81
C1Ø5: BPL $C1ØØ
C1Ø7: LDX # $Ø4
C1Ø9: LDY # $9B
C1ØB: NOP
C1ØC: DEY
C1ØD: BNE $C1ØB
C1ØF: CLC
C11Ø: ADC $D419
C113: BCS $C1ØØ
C115: DEX
C116: BNE $C1Ø9
C118: LDX # $ØØ
C11A: LDY # $99
C11C: NOP
```

```
C11D: DEY
C11E: BNE $C11C
C12Ø: LDA $D419
C123: STA $C2ØØ,X
C126: INX
C127: NOP
C128: NOP
C129: BNE $C11A
C12B: LDA # $ØØ
C12D: STA $C3ØØ
C13Ø: STA $C3Ø1
C133: LDX # $Ø7
C135: LDY # $2Ø
C137: LDA $C21C,Y
C13A: CMP # $81
C13C: BPL $C144
C13E: INC $C3ØØ
C141: JMP $C147
C144: INC $C3Ø1
C147: DEY
C148: BNE $C137
C14A: LDA $C3ØØ
C14D: CMP $C3Ø1
C15Ø: PLA
C151: ROR
C152: PHA
C153: LDA # $ØØ
C155: STA $C3ØØ
C158: STA $C3Ø1
C15B: LDA # $2Ø
C15D: CLC
C15E: ADC $C138
C161: STA $C138
C164: DEX
C165: BNE $C135
C167: LDA # $1C
C169: STA $C138
C16C: PLA
C16D: CLC
C16E: ROR
C16F: PHA
C17Ø: CMP # $ØØ
C172: BEQ $C17A
C174: JSR $E716
C177: JMP $C1ØØ
C17A: RTS
```

La versión en BASIC es la que sigue:

```
12ØØ FOR I=494Ø8 TO 4953Ø:
     READ A: POKE I,A: NEXT
121Ø DATA 173,25,212,2Ø1,
     129,16,249,162,4,16Ø,
     155,234,136,2Ø8,252,
     24,1Ø9,25,212
122Ø DATA 176,234,2Ø2,2Ø8,
     241,162,Ø,16Ø,153,234,
     136,2Ø8,252,173,25,
     212,157,Ø
123Ø DATA 194,232,234,234,
     2Ø8,239,169,Ø,141,Ø,
     195,141,1,195,162,7,
     16Ø,32,185
124Ø DATA 28,194,2Ø1,129,16,
     6,238,Ø,195,76,71,193,
     238,1,195,136,2Ø8,237,
     173,Ø
125Ø DATA 195,2Ø5,1,195,1Ø4,
     1Ø6,72,169,Ø,141,Ø,195,
     141,1,195,169,32,24,
     1Ø9,56
126Ø DATA 193,141,56,193,
     2Ø2,2Ø8,2Ø6,169,28,
     141,56,193,1Ø4,24,1Ø6,
     72,2Ø1,Ø,24Ø
127Ø DATA 6,32,22,231,76,Ø,
     193,96
```

BARRA DE LEDS

Otra aplicación interesante es utilizar el controlador de voz como medidor de intensidad sonora, a modo de «sonómetro». Un ejemplo de ello es como medidor de la potencia musical de cualquier amplificador. El programa desarrollado visualiza en la pantalla una doble barra de «leds», que se va iluminando al compás de la música o de la voz, del mismo modo que en los equipos musicales de calidad. El programa simula una columna doble de 20 LEDs, siendo los 5 últimos de color rojo y los restantes de color verde. En este caso deberá ponerse el micrófono cerca del altavoz de la fuente de sonido.

El programa se presenta también en ensamblador y con un cargador en BASIC. Para activarlo será necesario escribir la instrucción SYS 49152, y para detenerlo es necesario pulsar las teclas RUN/STOP y RESTORE.

El programa comienza limpiando la pantalla y poniéndola en negro. A continuación se dibujan los LEDs, que vienen representados por el signo menos. Hecho esto comienza la ejecución del programa principal, que se encarga de leer 20 veces el registro del conversor A/D y encender los LEDs en función de los datos leídos.

Si la intensidad sonora es demasiado fuerte la barra de LEDs marca siempre el valor máximo. En este caso es necesario separar el micrófono a una distancia conveniente para que las barras sigan correctamente la música.

Existe la posibilidad de realizar un medidor para equipos estereofónicos mediante dos controladores, uno para cada canal, y conectando el segundo al PIN 5, correspondiente a la entrada POT Y (posición 54298). El programa debe sufrir algunas modificaciones sencillas para este modo de funcionamiento.

PROGRAMA «BARRA DE LEDS» EN ENSAMBLADOR

```
CØØØ: JSR $E544
CØØ3: LDA # $2D
CØØ5: LDX # $14
CØØ7: STA $Ø4ØØ,X
CØØA: STA $Ø45Ø,X
CØØD: DEX
CØØE: BNE $CØØ7
CØ1Ø: LDA # $ØØ
CØ12: STA $DØ2Ø
CØ15: STA $DØ21
CØ18: LDX # $14
CØ1A: TXA
CØ1B: PHA
CØ1C: LDX # $Ø2
CØ1E: LDY # $AØ
CØ2Ø: DEY
CØ21: BNE $CØ2Ø
CØ23: DEX
CØ24: BNE $CØ1E
CØ26: PLA
CØ27: TAX
CØ28: LDA $D419
CØ2B: CMP # $81
CØ2D: BPL $CØ32
CØ2F: INC $C1ØØ
CØ32: DEX
CØ33: BNE $CØ1A
CØ35: LDX $C1ØØ
CØ38: LDY # $14
```

```
CØ3A: LDA # $ØØ
CØ3C: STA $D8ØØ,Y
CØ3F: STA $D85Ø,Y
CØ42: DEY
CØ43: BNE $CØ3C
CØ45: LDA # $Ø2
CØ47: CPX # $ØD
CØ49: BMI $CØ55
CØ4B: STA $D8ØØ,X
CØ4E: STA $D85Ø,X
CØ51: DEX
CØ52: JMP $CØ47
CØ55: LDA # $Ø5
CØ57: STA $D8ØØ,X
CØ5A: STA $D85Ø,X
CØ5D: DEX
CØ5E: BNE $CØ57
CØ6Ø: LDA # $Ø1
CØ62: STA $C1ØØ
CØ65: JMP $CØ18
```

La versión BASIC es la siguiente:

```
1Ø FOR I=49152 TO 49255:
   READ A: POKE I,A: NEXT
2Ø DATA 32,68,229,169,45,
   162,2Ø,157,Ø,4,157,8Ø,4,
   2Ø2,2Ø8,247,169,Ø,141,32,
   2Ø8
3Ø DATA 141,33,2Ø8,162,2Ø,
   138,72,162,2,16Ø,16Ø,136,
   2Ø8,253,2Ø2,2Ø8,248,1Ø4,
   17Ø
4Ø DATA 143,25,212,2Ø1,129,
   16,3,238,Ø,193,2Ø2,2Ø8,
   229,174,Ø,193,16Ø,2Ø,169,
   Ø
5Ø DATA 153,Ø,216,153,8Ø,
   216,136,2Ø8,247,169,2,
   224,12,48,1Ø,157,Ø,216,
   157,8Ø
6Ø DATA 216,2Ø2,76,71,192,
   169,5,157,Ø,216,157,8Ø,
   216,2Ø2,2Ø8,247,169,1,
   141,Ø
7Ø DATA 193,76,24,192
8Ø SYS 49152
```

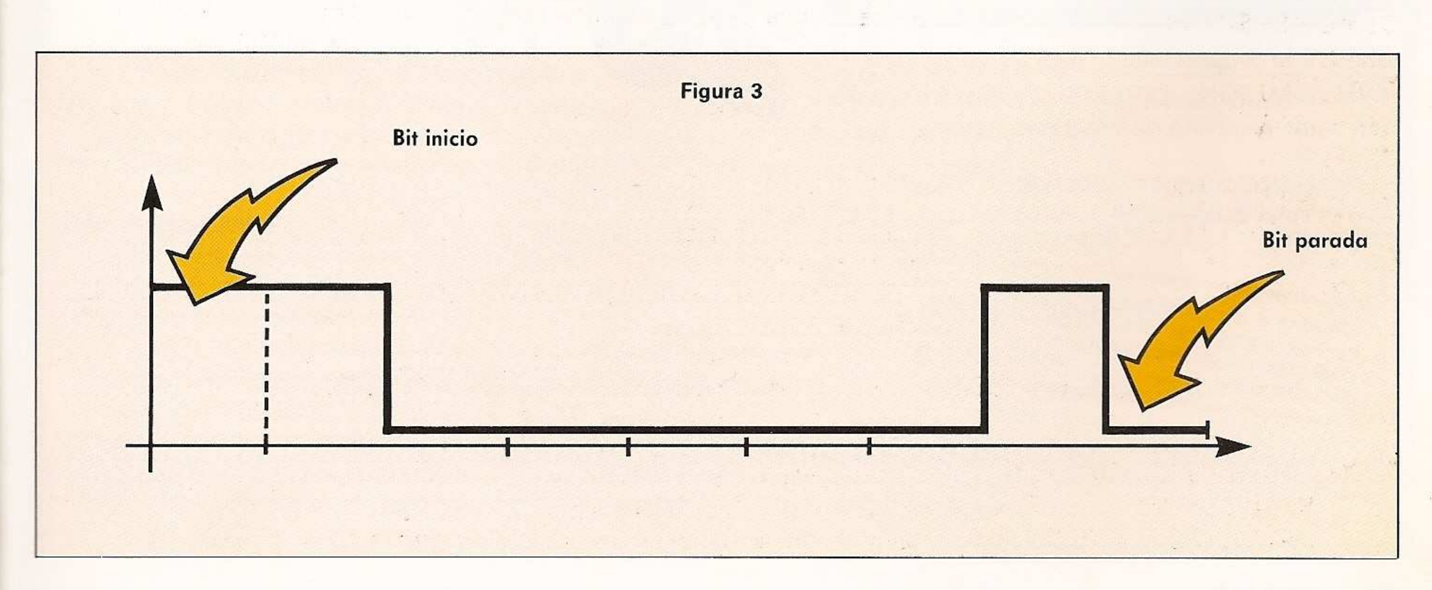
OTRAS APLICACIONES

Existe todavía otro grupo de importantes aplicaciones para el controlador: es el control de programas mediante la voz. Es posible gobernar un programa realizado adecuadamente pronunciando cualquier sonido ante el micrófono. Puede ser interesante este modo de control de programas para aquellas aplicaciones en las que, por el motivo que sea, no es posible manipular un teclado (ejemplo en programas educativos en niños de corta edad).

El método de programación consistirá en crear un menú rotativo que vaya presentando las diferentes opciones o respuestas posibles en pantalla de forma secuencial. Al llegar a la elegida bastará con emitir un sonido o una palabra cualquiera para que el ordenador reconozca la opción deseada.

El programa siguiente es un ejemplo de aplicación de esta técnica. En este caso se elige el color de la pantalla, pero evidentemente la misma estructura puede utilizarse para ejecutar cualquier otra tarea.

```
  5 A=1
 1Ø PRINT"(CLR)"
 2Ø PRINT"AMARILLO":
    PRINT
 3Ø PRINT"ROJO":
    PRINT
 4Ø PRINT"VERDE":
    PRINT
 5Ø PRINT"NEGRO"
 6Ø B=Ø:ON A GOTO 7Ø, 8Ø,
    9Ø, 1ØØ
 7Ø PRINT"(HOME)":PRINT"(RVS
    ON)AMARILLO":GOSUB15Ø:A
    =2:GOTO11Ø
 8Ø PRINT"(HOME)(3CRSDWN)
    (RVSON)ROJO":GOSUB
    15Ø:A=3:GOTO11Ø
 9Ø PRINT"(HOME)(5CRSDWN)
    (RVSON)VERDE":GOSUB
    15Ø:A=4:GOTO11Ø
1ØØ PRINT"(HOME)(7CRSDWN)
    (RVSON)NEGRO":GOSUB
    15Ø:A=1
11Ø IFB<>ØTHENONBGOSUB
    2ØØ,21Ø,22Ø,23Ø
12Ø GOTO1Ø
15Ø FORI=1TO2ØØ:C=PEEK
    (54297):IFC<128
```



```
    THENI=200:B=A
160 NEXT:RETURN
200 POKE 53280, 7: POKE
    53281, 7:RETURN
210 POKE 53280, 2: POKE
    53281, 2:RETURN
220 POKE 53280, 5: POKE
    53281, 5:RETURN
230 POKE 53280, 0: POKE
    53281, 0:RETURN
```

Y todavía pueden encontrarse otras muchas aplicaciones. Damos a continuación una serie de ejemplos:

—Niñera electrónica: es posible poner el micrófono del controlador en el cuarto de un bebé de forma que cuando éste llore el Commodore nos avise y le toque una canción de cuna.

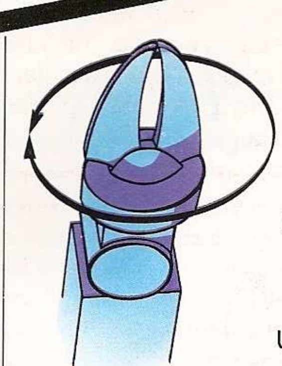
—Control de intrusos: cuando se detecte ruido en una habitación es posible utilizando el port del usuario conectar cualquier tipo de alarma o en una aplicación más sofisticada llamar a la policía y poner en marcha una grabación de forma automática.

—Registro de llamadas telefónicas: Situando el micrófono cerca del teléfono es posible registrar el número de llamadas y la hora en que fueron efectuadas.

—Encendido y apagado de aparatos: Cualquier aparato puede ser conectado mediante un relé al port del usuario y ser conectado y desconectado mediante, por ejemplo, una palmada. La alta sensibilidad del dispositivo permite captarla desde varios metros de distancia.



!PARTICIPA EN EL CONCURSO!



En INPUT estamos convencidos de que aún puedes hacer muchas más cosas con tu ordenador. Sin duda, muchos lectores estareis utilizando vuestro micro para funciones de lo más variadas, en unos casos; pintorescas, en otros; mientras que algunos listillos habrán podido utilizarlo para resolver tareas complejas. Es lógico, modificando programas y variando los periféricos nuestro ordenador puede prestar sus servicios en infinidad de facetas. INPUT quiere que esas aplicaciones y utilidades a las que has conseguido dedicar tu ordenador, sean conocidas por todos sus lectores y por eso ha organizado el «Concurso de Aplicaciones y Utilidades», en el que puede participar cualquiera de nuestros lectores.





UTILIDADES Y APLICACIONES: Si tu ordenador controla la calefacción de tu casa, gobierna un robot, dirige un pequeño negocio, organiza la maqueta de tu tren eléctrico, o cualquier cosa interesante u original; envíanos información gráfica y listados de tus programas, grabados en un cassette, diskette o microdrive.

Todo ello habrá de venir acompañado por un texto que aclare cuál es su objetivo, el modo de funcionamiento y una explicación del cometido que cumplen las distintas rutinas que lo componen. El texto se presentará en papel de tamaño folio y mecanografiado a dos espacios. No importa que la redacción no sea muy clara y cuidada; nuestro equipo de expertos se encargará de proporcionarle la forma más atractiva posible.

UN JURADO propio decidirá en cada momento qué colaboraciones reúnen los requisitos adecuados para su publicación, y evaluará la cuantía del premio en metálico al que se hagan acreedoras.

No olvideis indicar claramente para qué ordenador está preparado el material, así como vuestro nombre y dirección y, cuando sea posible, un teléfono de contacto. Entre todos los trabajos recibidos durante los próximos tres meses SORTEAREMOS:

- **Un premio de 50.000 ptas.**
- **Un premio de 25.000 ptas.**
- **Un premio de 10.000 ptas.**
  en material microinformático a elegir por los afortunados.

¡No os desanimeis!, por muy simples o complejas que puedan parecer vuestras ideas, todas están revisadas con el máximo interés.

INPUT COMMODORE
Aribau, 185. Planta 1.ª
08021 BARCELONA

NOTA: INPUT no se responsabiliza de la devolución del material que no vaya acompañado por un sobre adecuado con el franqueo correspondiente.

AZAR Y PROBABILIDAD

Tanto si estás interesado en estrategias de juego como en predecir situaciones de vida o de muerte, no puedes esperarlo todo del azar. Analiza más bien cómo puedes obtener información sobre el futuro, ¡y gana!

La fuerza de un ordenador reside en su capacidad para obedecer instrucciones de una forma repetitiva, precisa y rápida. Si lo comparamos con el proceso casi instantáneo del cerebro humano, la actuación de un ordenador es, desde luego, insignificante. El cerebro humano es único para hacer juicios y comparaciones de parámetros como la distancia, la velocidad o la intensidad de la luz.

Pero también estas funciones más sutiles fallan estrepitosamente cuando tratan de adivinar el resultado de un suceso. Y, sin embargo, es importante poder afirmar con seguridad que «a efectos de un seguro, se considera que una persona llega a vivir hasta los 65 o los 70 años», o bien que «no es probable que se dé un terremoto de gran alcance en España en lo que queda de siglo». Tales afirmaciones son normales en el lenguaje de la vida diaria (ponderamos el riesgo) y son también importantes para fines científicos, sociales o comerciales. Cuando utilizamos términos como *riesgo, pronóstico, duda, esperanza* y *posibilidad* estamos haciendo cálculos mentales de *probabilidades*.

LA PROBABILIDAD TAL COMO ES

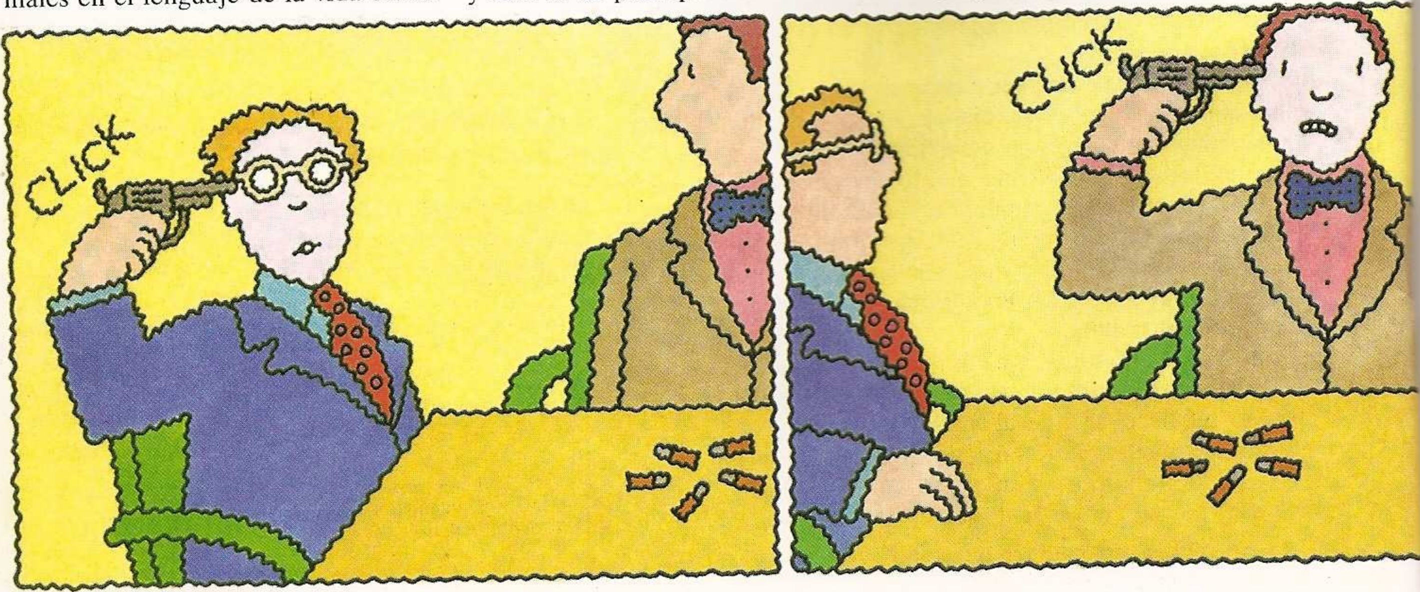
La probabilidad es la medida científica del azar, y se emplea para juzgar el posible resultado de un suceso. Se basa en la existencia de un número finito de resultados posibles, como pueden darse en los partidos de fútbol, el lanzamiento de una moneda, tirada de dados, reparto de cartas de baraja o el juego de las máquinas de frutas. Es claro que podemos medir o cuantificar los resultados, de forma que sucesos como las carreras de caballos o una competición de baloncesto constituyen la materia difícil de la probabilidad. Si tú dices «*Espero* ganar», lo que estás diciendo es que existe una elevada *probabilidad* de que te sonría la suerte.

Muchos se fían sólo de la intuición, y ésta es su principal herramienta de cálculo de probabilidades. Pronostican, y basta. Pero tú puedes hacerte una idea más precisa de ciertas situaciones en las que interviene el azar si te detienes a examinar cuáles son los sucesos posibles, y aunque nunca tendrás la certeza absoluta estarás en mejores condiciones que el que emite su pronóstico poco o nada *documentado* sobre el tema.

PROBABILIDAD E INFORMATICA

¿Y qué tienen en común la probabilidad y la informática? Pues, hablando en castizo, *la tira*. Aunque el tipo de probabilidad antes mencionada es bastante amplia y depende de muchos factores, es posible deducir reglas matemáticas para ciertos tipos de sucesos que nos permitan predecir el resultado más verosímil con un cierto margen de seguridad.

Los ordenadores pueden ser útiles por dos vertientes. Por un lado, pueden utilizarse para simular el resultado mismo; es mucho más fácil programar un ordenador para que te tire un dado dos mil veces que hacerlo tú con tu propia mano. Por otro lado, si sabes



- ¿QUE ES PROBABILIDAD?
- AZAR, PROBABILIDAD Y FRECUENCIA
- PROGRAMA PARA LANZAR UNA MONEDA
- PROBABILIDAD DE VARIOS RESULTADOS
- TRIANGULO DE PASCAL
- DISTRIBUCION DE FRECUENCIAS
- PREDICCION DE RESULTADOS

las fórmulas del resultado esperado, puedes emplear el ordenador para que te calcule también el resultado.

Según sean tus intereses, esto puede reducirse a un mero ejercicio teórico, o convertirse en la base de muchas aplicaciones prácticas. Considera sólo el mundo de los juegos de azar, en los que, por ejemplo, tu apuesta depende de la verosimilitud de determinados resultados. Pero también se podría escribir un programa para determinar la probabilidad de que llueva en un día determinado (¡o la probabilidad de que acontezca una erupción volcánica en la península!). De momento ciñámonos a la teoría. Más adelante INPUT te enseñará a montar algunas de esas aplicaciones prácticas.

MEDIR LA PROBABILIDAD

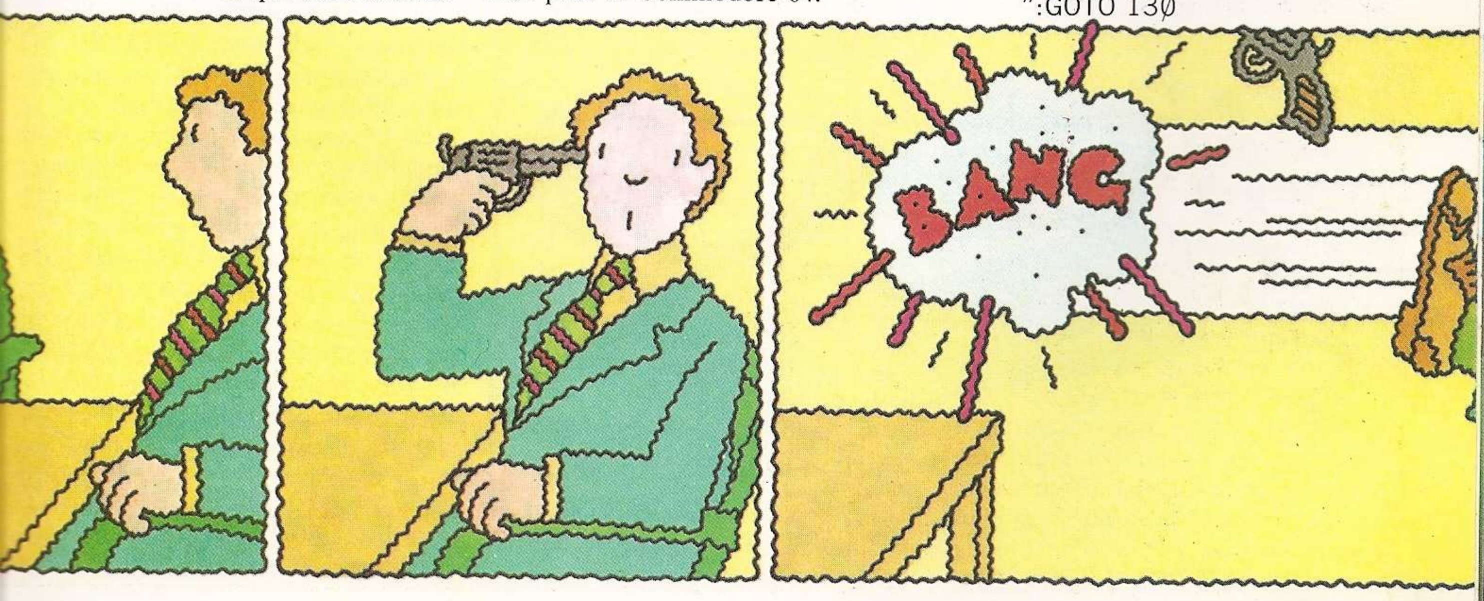
La teoría de la medición de la probabilidad pide que conozcas de antemano el número de sucesos posibles de un determinado experimento, y la frecuencia con que suele ocurrir cada uno de ellos. La probabilidad de que ocurra un determinado suceso es igual al número de veces que suele suceder (su frecuencia) partido por el número total de sucesos posibles.

Si un suceso es seguro que ocurrirá, su probabilidad será, según lo dicho, igual a 1. Está claro que si siempre ocurre, su frecuencia es igual al número de resultados posibles, es decir, dividimos un número por sí mismo, luego es igual a 1. Conclusión: la probabilidad más alta de un suceso vale 1. Y para muchos sucesos posibles e independientes, si sumamos sus respectivas probabilidades nos dará también 1.

Uno de los métodos más sencillos y antiguos de razonamiento probabilístico consistía en lanzar una moneda al aire y predecir su resultado. Dado que una moneda sólo ofrece dos resultados, intuitivamente podemos deducir que si la lanzamos al aire un buen número de veces, obtendremos cara la mitad de las veces y cruz la otra mitad, dando por despreciable la bajísima probabilidad de que la moneda se quede de canto. Para ilustrar este método, introduce y ejecuta este primer programa.

Necesitas el cartucho BASIC de Simon para tu Commodore 64.

```
1Ø PRINT "[SHIFT+CLR/
   HOME][CTRL+7]ENTRAR
   OPCION (1-4)"
2Ø POKE 5328Ø, 3:POKE
   53281, 1
3Ø INPUT X:PRINT "SHIFT+CLR/
   HOME]":IF X<1 OR X>4
   THEN1Ø
4Ø ON XGOTO 7Ø; 7Ø, 18Ø; 46Ø
5Ø REM....PROBABILIDAD
6Ø REM......LANZAMIENTO DE
   UNA MONEDA
7Ø H=Ø:T=Ø
8Ø PRINT "[CLR/HOME][CTRL
   +9[ PULSA -ESPACIO- PARA
   LANZAR LA MONEDA. "
9Ø PRINT "[CLR/HOME][
   4*CRSR ABAJO]CARA :- Ø"
   :PRINT "CRUZ :-Ø"
1ØØ GET A$:IF A$<>" "
    THEN1ØØ
11Ø IF X=2 THENFOR N=1 TO
    1ØØ
12Ø IF INT (RND(1)*2)+1=1
    THENH=H+1:PRINT "[CLR/
    HOME][ 2*CRSR
    ABAJO][CTRL+9]CARA
    ":GOTO 13Ø
```

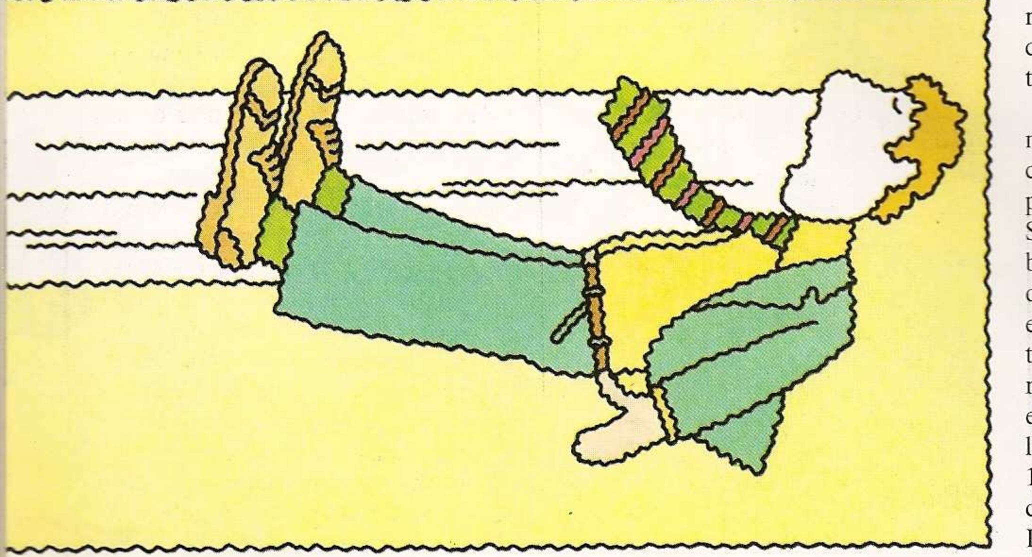


```
125 T=T+1: PRINT "[CLR/
    HOME][ 2*CRSR
    ABAJO][CTRL+9]CRUZ "
130 PRINT "[CLR/HOME][
    4*CRSR ABAJO]"; TAB(7);
    H:PRINT TAB(7); T
140 IF X=1 THENGET A$:IF
    A$<>" " THEN130
150 IF X=1 THEN120
160 NEXT N:END
```

Este programa será desarrollado a lo largo del artículo. Cuando lo ejecutes (RUN) tendrás que introducir un número para seleccionar una prueba. En este momento sólo has entrado la primera de las pruebas, luego escribe un 1; estás ahora en condiciones de lanzar la moneda apretando la barra espaciadora o SPACE. La esencia del programa está en la línea 120 que proporciona unos (caras) y ceros (cruces) al azar. Cuando sale cara, la línea 120 imprime cara, y cuando sale cruz la línea 125 imprime (PRINT) cruz. Esta misma línea tiene un contador del número de caras y cruces que van saliendo en los distintos lanzamientos.

Con un número escaso de lanzamientos obtendremos valores muy distintos de cara y de cruz, pero si aumentamos éstos veremos cómo tales valores se van aproximando cada vez más, hasta distribuirse por igual la mitad de los lanzamientos: o sea, el 50 % de las veces salió cara y el 50 %, cruz. Compruébalo ejecutando el programa después de haber entrado el 2, para seleccionar la segunda prueba. Esta vez, cuando aprietes la barra o SPACE, la línea 110 establece un bucle con el que se lanzará 100 veces la moneda. Verás que en pantalla sale para CARA y para CRUZ un número muy próximo a 50. Si quieres, cambia el 100 de la línea 110 por el 1000 y vuelve a ejecutar el programa, introduciendo como antes el número 2 para que repita la segunda prueba. Resultado: CARA y CRUZ reciben casi 500 por igual.

Desde luego, es posible obtener cara en todos los lanzamientos, nadie puede negar esto, pero es probable obtener cara sólo en la mitad de dichos lanzamientos. Anótate esto bien cuando examines más de un experimento. Mucha gente cree que si seguimos lanzando una moneda después de haber obtenido diez caras, es mayor la probabilidad de obtener cruz ahora que al principio. Pero esto no es verdad. Los sucesos pasados no influyen sobre el suceso futuro de obtener una cruz o una cara en el siguiente lanzamiento. Pero si vas a lanzar 11 monedas de golpe, deberás saber que la probabilidad de obtener 11 caras es más pequeña que la de obtener diez caras y una cruz, y es todavía más probable que obtengas un número casi igual de caras que de cruces.



EXPERIMENTOS MULTIPLES

Cuando existen varios experimentos, es necesaria alguna información más para poder predecir la probabilidad de cada suceso. Un dato informativo esencial es el total de todos los sucesos posibles. Por ejemplo, si lanzas dos veces una moneda, tres son los sucesos posibles: dos caras, cara y cruz, dos cruces. Podemos pensar que cada uno de estos sucesos puede ocurrir la tercera parte de las veces. De hecho, las probabilidades son: dos caras (1/4), dos cruces (1/4) y cara y cruz (1/2). Para entender esta última probabilidad de 1/2 deberás utilizar otra información más: el número de ocurrencias de cada suceso. Cara y cruz ocurre dos veces, ya que hay *dos* maneras de obtener este resultado (cara y cruz, o bien cruz y cara) con lo que tenemos un total de cuatro sucesos, de los cuales tres son distintos.

En la práctica, hay dos trucos matemáticos que te ahorran el esfuerzo de determinar el número de sucesos. Son el *teorema del binomio* y el *triángulo de Pascal*. Binomio significa «de dos términos». Si un experimento sólo tiene dos sucesos posibles y conoces la probabilidad de cada uno de ellos, nos serviremos del teorema del binomio para obtener las probabilidades.

El teorema del binomio nos indica lo que se debe esperar de las pruebas que se repiten de un experimento con dos sucesos. Llamemos P la probabilidad de un resultado y Q la del otro resultado (nota que $P + Q$ vale 1, como ya sabes). Y llamemos, por último, N al número de sucesos.

En el ejemplo de lanzar una moneda, P es la probabilidad de que salga cara, y Q de que salga cruz. Entonces, para un lanzamiento, P y Q valen 1/2. Según el teorema del binomio, la probabilidad en un experimento que se da dos veces, es la probabilidad cuando el experimento sólo se da una vez multiplicada por sí misma. En general, la regla dice que la probabilidad hay que elevarla a N. Así para dos caras en dos lanzamientos tendremos $P \uparrow N = 1/2^* 1/2 = 1/4$. Hay una posibilidad sobre cuatro de obtener dos caras a la vez. Igualmente, la probabilidad de obte-

ner cinco caras a la vez, será P ↑ N, o sea 1/2 ↑ 5, es decir, 1/32.

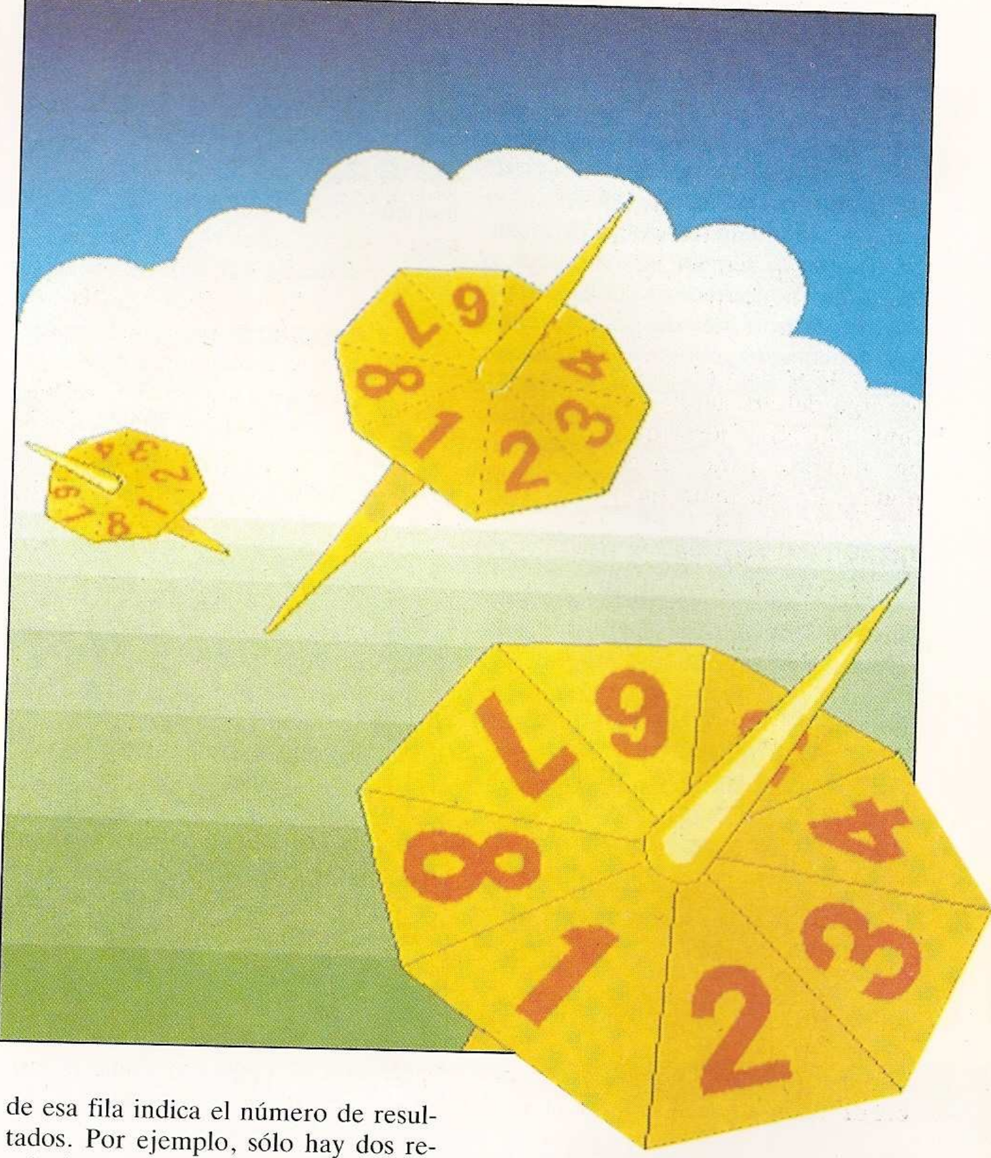
Como verás más adelante, este método se puede emplear para calcular la probabilidad de cualquier suceso cuando sólo existen dos resultados posibles, los sucesos sí/no o bien cara/cruz. Pero ¿y el caso de obtener tres caras y dos cruces después de cinco lanzamientos de una sola moneda? Para responder a esto necesitamos un planteamiento más complejo.

El triángulo de números, ideado por el matemático francés Blas Pascal, sirve en muchas aplicaciones matemáticas, y entre ellas la que nos ocupa. Ofrece todos los resultados posibles de cualquier experimento con dos resultados, y puede ser dispuesto como una sucesión de filas de números. Las primeras siete filas son:

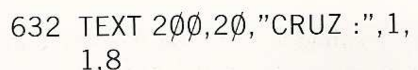
Fila 0	1
Fila 1	1 1
Fila 2	1 2 1
Fila 3	1 3 3 1
Fila 4	1 4 6 4 1
Fila 5	1 5 10 10 5 1
Fila 6	1 6 15 20 15 6 1

Para construir un triángulo como éste, escribe primero las dos primeras filas (fila 0 y fila 1), que son fáciles de recordar. La fila 2 comienza con un 1 a la izquierda y acaba con otro 1 a la derecha de la fila 1. El número del medio (2) se obtiene sumando los números de la fila anterior (1 + 1). Igualmente, la fila 3 se obtiene colocando sendos unos a la derecha y a la izquierda, y los restantes números son la suma de los números de la fila anterior (fila 2): 1 + 2 y 2 + 1. Así se han obtenido las demás filas, como la fila 6: 1 — 1 + 5 — 5 + 10 — 10 + 10 — 10 + 5 — 5 + 1 — 1. Y así podrías enriquecer este triángulo con todas las filas que desees, aunque el triángulo te ocupará entonces mucho espacio.

El *triángulo de Pascal* (también llamado de Tartaglia), te ofrece la información que necesitas, cuando se trata del lanzamiento de varias monedas al mismo tiempo (o de una moneda que se lanza varias veces). El número de monedas nos dice la fila que hay que inspeccionar; el número de elementos de esa fila indica el número de resultados. Por ejemplo, sólo hay dos resultados para una sola moneda (fila 1: el 1 y el 1) y siete resultados posibles para seis monedas (fila 6: 1, 6, 15, 20, 15, 6 y 1). La suma de los números de la fila proporciona el número total de resultados (2 para una moneda, 4 para dos monedas, etc.). Cada número de la fila es una probabilidad. Por ejemplo, en la fila 2, el primer número (1) es la probabilidad de obtener dos caras, el segundo número (2) es la de cara y cruz, y el tercer número (1) es la de dos cruces. Naturalmente estos números nos dan la frecuencia que hay que dividir por el número total de resultados posibles (en nuestro caso, cuatro) si queremos hallar la probabilidad propiamente dicha. Observa cómo el resultado de sumar los números de cada fila da siempre una potencia del dos (1, 2, 4, 8, 16). Esto se explica porque, para cualquier experimento, sólo existen dos sucesos posibles.



Ya puedes ir percatándote de lo útil que resulta este método cuando se trata de calcular las probabilidades resultantes de lanzar, por ejemplo, 30 monedas al aire, pero sería algo aburrido intentar resolverlo mediante la construcción de un triángulo de 30 filas, aparte del espacio que esto te iba a ocupar. Existe, en su lugar, un método gráfico que nos permite encarar tales casos, y aquí es donde puedes hacer intervenir a tu ordenador.

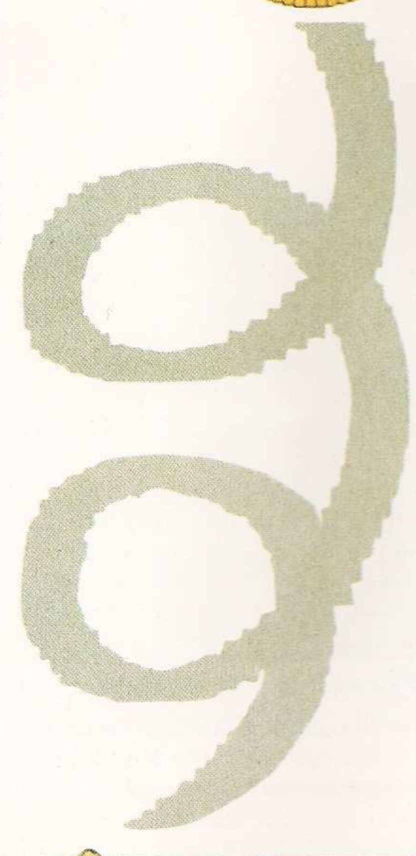


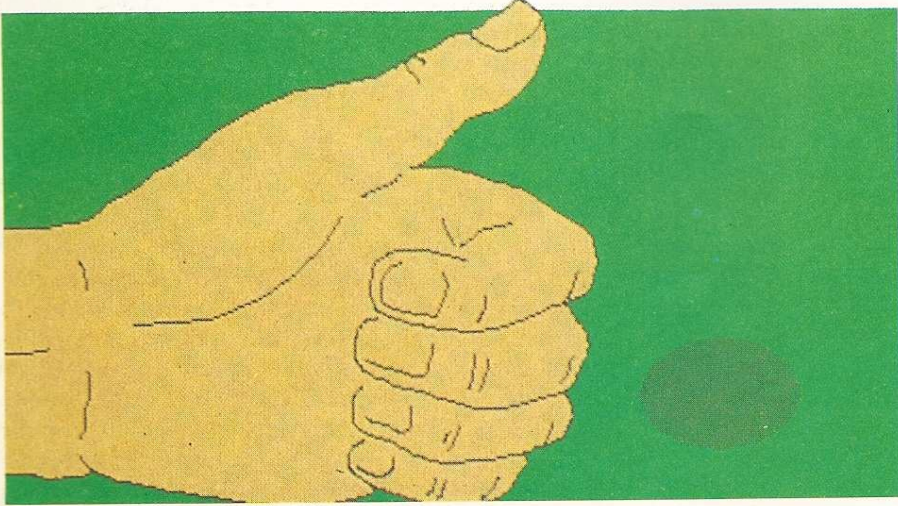
CURVAS DE DISTRIBUCION

Cuando existen muchos sucesos, y sus probabilidades no parecen tan evidentes, a menudo pueden obtenerse buenos resultados mediante el trazado de una curva de distribución. Ésta se traza mediante la frecuencia de los sucesos que se tengan tabulados: es la distribución de frecuencias. Como sucede con todo método gráfico, de un vistazo puede obtenerse la mayor parte de la información necesaria. Si, por ejemplo, vas a jugar al lanzamiento de una moneda 30 veces (que es lo mismo que lanzar 30 monedas de una sola vez), puedes visualizar el número de caras (o cruces) que salen en cada juego de 30 lanzamientos. Para ver el resultado, escribe la siguiente sección del programa, sin borrar la anterior sección:

```
17Ø REM......MAXIMOS
    ALEATORIOS
18Ø HIRES Ø,1:FOR X=Ø TO
    3ØØ STEP2Ø
19Ø GM=Ø:GOSUB 61Ø
2ØØ FOR Y=Ø TO H*6
    STEP6
21Ø TEXT X, 200-Y, "*", 1,
    1,8
22Ø NEXT Y, X
23Ø GOTO 23Ø
61Ø REM.....LANZAMIENTO
62Ø H=Ø:T=Ø
63Ø TEXT 2ØØ,1Ø,"CARA:",1,
    1,8
632 TEXT 2ØØ,2Ø,"CRUZ :",1,
    1,8
64Ø IF GM<>Ø THENTEXT Ø,Ø,
    "JUEGOS:",1,1,8:TEXT
    1ØØ, Ø, "CARAS DE 3Ø
    JUEGOS:",1,1,8
65Ø FOR TS=1 TO 3Ø
66Ø IF INT (RND(1)*2)+1=1
    THENH=H+1:GOTO 67Ø
665 T=T+1
67Ø TEXT 263,1Ø,STR$(H),1,1,
    8
672 TEXT 263,2Ø,STR$(T),1,
    1,8
674 TEXT 263,1Ø,STR$(H),Ø,1,
    8
676 TEXT 263; 2Ø, STR$(T), Ø,
    1, 8
678 NEXT TS
68Ø RETURN
```







Ejecuta el programa, entrando ahora el 3 para seleccionar la tercera prueba. Verás un gráfico con una sucesión de puntos que ascienden hasta varios puntos elevados sobre la pantalla. Ésta es una de las muchas figuras posibles en este tipo de análisis. Los puntos elevados son el número de caras que se obtienen de 30 lanzamientos y se trazan a lo largo del eje Y, extendidas a lo largo del eje X. Observa que hay más cumbres altas que bajas. La razón de esto es que hay más probabilidad de obtener 15, o bien entre 12 y 17 caras es mucho más elevada que la de obtener un número menor o mayor de caras. Esto mismo es lo que puede verse en los números del triángulo de Pascal, con valores mayores justo para los números de en medio.

La línea 180 establece un bucle para extender los puntos elevados a lo largo del eje X. La variable GM se inicializa a cero y señala el número de juegos de 30 lanzamientos (línea 190) y se llama a una rutina (líneas 610 a 680) donde se realiza cada juego de 30 lanzamientos. Esta rutina emplea los elementos de la segunda prueba, pero lanza la moneda «electrónica» 30 veces, en lugar de 100. Cuando ejecutes este programa verás que las letras CARA y CRUZ aparecen en la esquina superior derecha de la pantalla. Una vez realizados los 30 lanzamientos, el número de caras que se acumulan en la rutina se someten a una escala en la línea 200 y se trazan (PLOT) en la línea 210 como coordenadas de Y. Para

sacar todo el jugo de este estudio, precisas ordenar la información de modo que obtengas una de las más conocidas curvas estadísticas: la de distribución normal. Escribe estas pocas líneas y verás la curva de que te hablamos:

```
450 REM........DISTRIBUCION
    NORMAL
460 HIRES Ø,1:DIM G(3Ø)
470 LINE Ø,Ø,Ø,2ØØ,1:LINE Ø,
    2ØØ,32Ø,2ØØ,1
480 GOSUB 56Ø
485 GET A$:IF A$<>" "
    THEN485
490 END
560 REM GR
565 DEF FN N(X)=1/(EXEC
    *1.4142*2.718↑((X↑2)/2))
570 FOR X=1 TO 32Ø
580 PLOT X,199-(FN N ((X-16Ø)/
    24)*53Ø),1
590 NEXT X:RETURN
```

Ejecuta el programa y entra un 4 para ver una curva ideal del tipo distribución normal. La línea 460 da dimensión a una tabla, necesaria más adelante para guardar en ella la cuenta de caras obtenidas. La línea 470 dibuja dos ejes de coordenadas X-Y, y la línea 480 llama a una rutina para que dibuje la curva. Este rutina emplea una función matemática (línea 580) para dibujarla, lo cual explica su forma ideal: une todos los puntos para obtener una curva bien alisada. La función está definida en la línea 560. No pulses todavía ninguna tecla, pues la rutina está incompleta y te dará error.

Muy rara vez puede obtenerse una curva perfecta al representar la información de los datos disponibles. Esto es de esperar, pues estás tratando probabilidades, no certezas. La probabilidad de un suceso (como, por ejemplo, la de tener lluvias torrenciales durante el período monzónico en la India) es elevada, pero han existido períodos en que lo que se ha dado es sequía en lugar del esperado diluvio. La siguiente prueba ilustra perfectamente este punto. Lo que hace es repetir el experimento un buen número de veces y representar gráficamente los resultados. Escribe la segunda parte de nuestra cuarta prueba:

```
490 FOR GM=1 TO 2ØØ
500 GOSUB 61Ø
502 TEXT 5Ø, Ø,STR$(G1),Ø,1,8
504 TEXT 263,Ø,STR$(G2),Ø,1,8
510 G(H)=G(H)+1
520 PLOT H*1Ø+1Ø,2ØØ-
    G(H)*4,1
530 TEXT 5Ø,Ø,STR$(GM),1,1,
    8:G1=GM
532 TEXT 263,Ø,STR$(H),1,1,
    8:G2=H
540 NEXT GM
550 GOTO 55Ø
```

Ejecuta ahora la cuarta prueba de nuevo. Una vez dibujada la curva ideal, pulsa la barra espaciadora o SPACE para iniciar el lanzamiento. Observa ahora la serie de puntos que va «creciendo» hasta llenar el espacio dentro de la curva. Cuando se completa la prueba, se habrán dibujado 200 puntos (PLOT de la línea 490). En algunos micros, y el Commodore 64 es uno de ellos, el tiempo que se invierte en este experimento es de varios minutos. Ésta es la razón por la que se ha escogido este número de 200 en lugar de otro más alto, como el 500 en la línea 490.

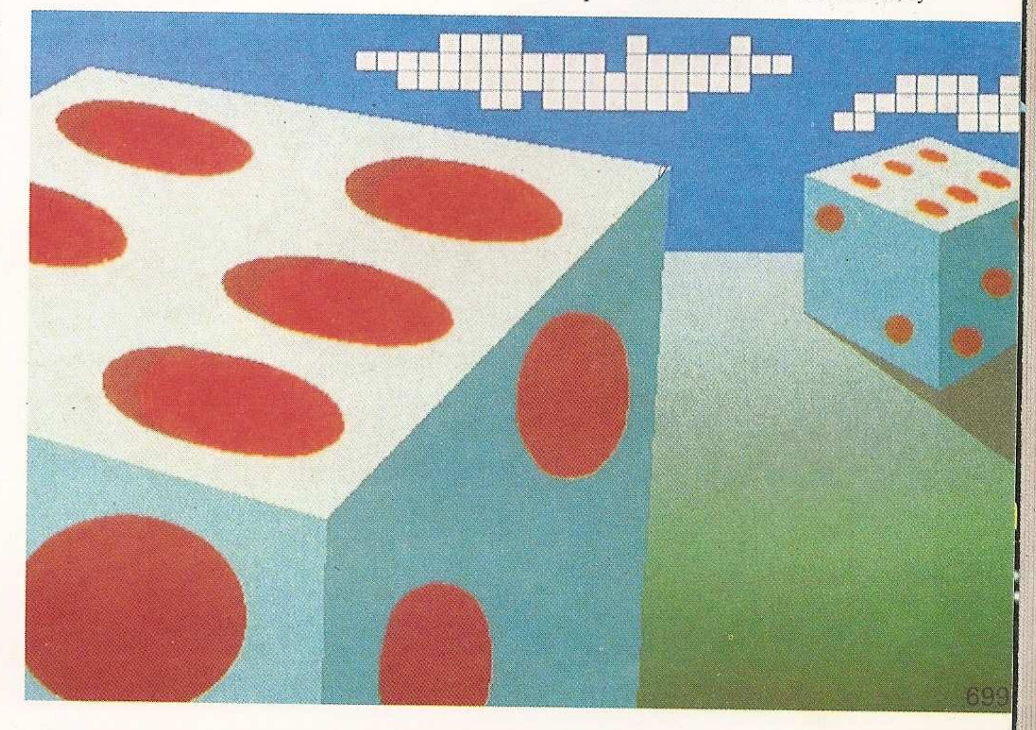
Esta parte del programa llama a la rutina (línea 500) que lanza la moneda 300 veces de modo que, al igual que sucedía en la tercera prueba, los lanzamientos son iluminados en la parte superior derecha de la pantalla. Cada serie de 30 lanzamientos es un juego, y puede dar un número cualquiera de caras entre 0 y 30.

Mediante la tabla, la línea 510 guarda la cuenta de los resultados de cada juego. Por ejemplo, cada vez que el resultado de un juego es 11 caras, la casilla G (11) se incrementa en una unidad, y lo mismo pasa con la G (15) si el resultado fue de 15 caras. Al inicio todas las casillas están a 0.

Tras cada juego, la línea 520 somete a escala el valor de cara (el número de caras en 30 lanzamientos) para obtener las coordenadas X e Y. La siguiente vez que sucede el mismo resultado se traza un punto en la misma posición de X, pero una unidad más adelante en el eje de Y. La línea 530 guarda la cuenta de H para cada juego, y también la cuenta del número de juegos efectuados.

EMPLEO DE LA CURVA

Ejecuta la cuarta prueba unas cuantas veces para ver cómo varía el perfil de los puntos dentro de la curva, y



después haz lo mismo pero con valores finales de GM más pequeños (línea 490). Aun sin la curva ideal, pronto podrás imaginar una curva idealizada a través de los puntos elevados. En la práctica, sin embargo, la inversa de este proceso imaginario es lo que resulta de un inmenso valor: sabiendo el perfil de una curva, predecir los resultados de un experimento futuro.

El valor de cara en un punto central es de especial interés. Es la denominada *media matemática*, de los 31 valores cara posibles a lo largo del eje X. En este caso, es 15. La media se identifica con el punto máximo de la curva. Es el valor con mayor probabilidad, pero no es útil por sí mismo. Se puede decir que 15 es lo más probable, pero también diremos que 14 y 16 son casi tan probables. Hay varios valores comunes en torno al máximo, y esto sí que es útil: saber su grado de dispersión. Por tanto, la media se utiliza para especificar otro parámetro de vital importancia: la desviación típica, o la medida de la dispersión. La fórmula de la desviación típica es complicada, pero no sin razón. Una vez calculado este parámetro, puedes asignar probabilidades a todos los puntos de la curva.

La desviación típica es una medida de la variación de los valores a ambos lados de la media. Por ejemplo, una sección de la curva con una desviación de 1,96 en cada lado de la media incluirá el 95 % de los resultados. Si amplías la variación típica a 2,58 la curva incluirá el 99 % de los resultados. Si empleas un software de estadísticas comerciales, la obtención de la desviación típica es superfácil.

UN CASO CON SEIS SUCESOS

Hay muchos ejemplos de experimentos que tienen más de dos sucesos posibles, cosa que no ocurría en el lanzamiento sencillo de una moneda. En estos casos, determinar la probabilidad de un suceso no es tan fácil, ni se soluciona recurriendo a la respectiva fila del *triángulo de Pascal*. Por ejemplo, en el caso de un dado, al tirarlo puedes obtener seis posibles resultados. Si el dado no está cargado, los seis resultados tienen la misma probabilidad. ¿Y los posibles resultados de tirar dos dados? Se pueden enumerar mediante una tabla, pero siempre has de notar que cuantos más resultados posibles haya más complicada es la tarea de determinarlos.

He aquí una tabla de todos los resultados posibles al tirar un par de dados a la vez:

		Valor del primer dado					
		1	2	3	4	5	6
	1	2	3	4	5	6	7
Valor	2	3	4	5	6	7	8
del	3	4	5	6	7	8	9
segundo	4	5	6	7	8	9	10
dado	5	6	7	8	9	10	11
	6	7	8	9	10	11	12

Como puedes ver, hay 36 posibles resultados (seis filas por seis columnas) aunque sólo 11 son diferentes. Pero hay más cosas que se deducen de esta tabla. Sólo hay una posibilidad contra 36 de obtener la suma más baja o la suma más alta (2 y 12 sólo aparecen en la tabla una vez), mientras que hay seis posibilidades contra 36 (o sea 1/6) de obtener 7 de suma. Y hay también 6 posibilidades contra 36 de obtener un doble. Es lo que ofrece la diagonal que va desde el número 2 (doble de doses) hasta 12 (doble de seises).

TIRADA MULTIPLE DE DADOS

Combinando el teorema binomial y esta tabla, puedes calcular las probabilidades de muchos experimentos. Un buen ejemplo de tirada múltiple de dados se encuentra en el juego del *Monopoly*. Si caes en la cárcel, tienes tres oportunidades para obtener un doble, de lo contrario has de pagar una fianza. Intuitivamente parecería que tienes un 50 % de probabilidades de salir de la cárcel (3 tiradas, con un 1/6 de probabilidad cada vez), pero no es así. En la tabla puedes ver que la probabilidad de que *no* obtengas un doble en cada tirada es de 30/36 (5/6). Con el teorema del binomio, puedes ver que la posibilidad de no obtener un doble las tres veces juntas es 5/6 elevado a 3, o sea 125/216. Lo que da alrededor de un 58 % de posibilidad de fracaso, por tanto si tu dinero te lo permite, mejor que no tientes a la mala suerte y pagues la fianza para librarte de la cárcel.

GANADORES DE LOS MEJORES DE INPUT COMMODORE

En el sorteo correspondiente al número 19 entre quienes escribisteis mandando vuestros votos a LOS MEJORES DE INPUT han resultado ganadores:

NOMBRE	LOCALIDAD	JUEGO ELEGIDO
Carlos Santolaya García	Sta. M.ª de Montcada (Barcelona)	GAUNTLET
J. Ignacio Luengo Crespo	Alcorcón (Madrid)	WORLD GAMES
Jesús Losada Prieto	Benavente (Zamora)	DAN DARE
David Tibau Llinas	Lloret de Mar (Girona)	ASTERIX
Andrés López Pascual	San Juan (Alicante)	EL NINJA
Gabriel Maldonado Suárez	Almería	TWO ON TWO
Hilario García Ostos	Peñarroya (Córdoba)	TWO ON TWO
Santiago Delgado Llopis	Madrid	EXPRESS RAIDER
Mariano Alcázar Cano	Elche (Alicante)	MARBLE MADNESS
Rubén Rodríguez de Torres	Vigo (Pontevedra)	LCP


forum
JUVENIL
LA GUERRA DE LAS GALAXIAS
CADA MES EN TU QUIOSCO
COMICS forum
¡DOS COMIC-BOOKS USA AL PRECIO DE UNO!
¡EL JUEGO MORTAL DE DARTH VADER!
Nº 7
140 PTAS

LOS MEJORES DE INPUT

MAYO 1987

commodore

PUESTO	TITULO	PORCENTAJE
1.º	Comando	17,3 %
2.º	Green Beret	15,2 %
3.º	Uridium	12,4 %
4.º	Sky Fox	12,2 %
5.º	Rambo	11,1 %
6.º	Dragon's Lair	10,8 %
7.º	Spindizzy	6,1 %
8.º	Leader Board Golf	5,2 %
9.º	Dentro del Laberinto	4,8 %
10.º	Marble Madnes	4,8 %
		100,0 %

Para la confección de esta relación únicamente se han tenido en cuenta las votaciones enviadas por nuestros lectores de acuerdo con la sección «LOS MEJORES INPUT».

ELIGE TUS PROGRAMAS

Hemos pensado que es interesante disponer de un *ranking* que ponga en claro, mes a mes, cuáles son los programas preferidos de nuestros lectores. Para ello, es obligado preguntaros directamente y tener así el mejor termómetro para conocer vuestras preferencias. Podéis votar por cualquier programa aunque no haya sido comentado todavía en INPUT.
El resultado de las votaciones será publicado en cada número de INPUT.
Entre los votantes sortearemos 10 cintas de los títulos que pidáis en vuestros cupones.
Nota: No es preciso que cortéis la revista, una copia hecha a máquina o una simple fotocopia sirven.

Enviad vuestros votos a: LOS MEJORES DE INPUT Aribau, 185. Planta 1. 08021 Barcelona

COM. N.º 19

1.er Título elegido

2.º Título elegido

3.er Título elegido

Programa que te gustaría conseguir ACE OF ACE

Qué ordenador tienes COMMODORE-64C

Nombre MOISES AUGUSTO

1.er Apellido COLQUEHUANCA

2.º Apellido VILLAR

Fecha de nacimiento 22-MARZO-1972

Teléfono

Dirección JIRON ENRIQUE LOPEZ ALBUJAR #345 (La Capilla)

Localidad JULIACA

Prov. SN ROMAN-PERU

SOFTACTUALIDAD

INDOOR SPORTS

El juego que os comentamos es un simulador; pero no de vuelo, se trata de una simulación de deportes de sala como los bolos, los dardos o el críquet.

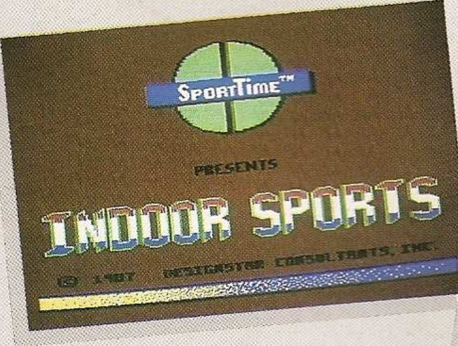
Posiblemente, si este juego se hubiese producido en nuestro país simularía otro tipo de juegos que tienen muchos más seguidores que los que aquí comentamos.

La elección del deporte de sala que quieres practicar se efectúa de forma muy original. Los gráficos son simpáticos y cuentan con una buena animación. Por ejemplo, al jugar a los bolos la escena se divide en dos partes, una, la primera, en la que elegimos la trayectoria y el ángulo de tiro y, otra, la segunda, en la que presencias el impacto sobre los bolos.

LEADER BOARD II

El éxito alcanzado por las diferentes versiones de este juego en los distintos sistemas ha convencido a ACCESS de la necesidad de realizar un *remake* de su celebérrimo juego de golf.

Sin embargo, no se trata de una simple adaptación del anterior pues, respecto al anterior, incluye incontables



e interesantes mejoras. Desde la posibilidad de cambiar los hoyos, los recorridos e, incluso, el paisaje. Incluso se aprecia como los árboles impiden la perfecta visión del recorrido de la pelota, lo que le da una proximidad con el juego real muy interesante.

Los campos también incorporan auténticos bunkers. Caso de que tus pelotas caigan en ellos, vas a pasarlo bastante mal hasta que consigas salir.

Además de estos detalles también destacan la posibilidad de graduar la fuerza con la que golpear la pelota, o los cambios de trayectoria de la pelota según cuales sean las características de la zona que esté atravesando.



PRINT SHOP COMPANION

Los habituales de INPUT ya sabéis que muchas veces nos gusta comentar programas que aunque no sean los clásicos juegos consideramos puede ser interesante que los lectores conozcan su existencia. Hace poco publicamos un artículo sobre software para impresoras.

En él comentábamos un programa de la casa BERKELEY, muy parecido al PRINT MASTER de BORDERBUND. Finalmente, esta última acabó denunciando por «plagio» a la primera. Denuncia que ganó.

Por lo que pudiera suceder BORDERBUND decidió ampliar considerablemente su programa. El resultado de esa decisión es el PRINT SHOP COMPANION. En él se incorporan una docena más de nuevos tipos de letras, un generador de



FONTS (tipos), un generador de fondos de papel, un generador de RECUADROS, de FILLS y, por último, un generador de gráficos que permite recuperar los dibujos generados con DODDLE y algunas más.

DRO AMPLIA HORIZONTES

Parece ser que la prestigiosa firma australiana Melbourne acaba de ser comprada por la destacada casa inglesa Mastertronic.

Aunque el precio no nos ha sido comunicado parece ser que la suma ha sido muy, muy sustanciosa, recordad que algunos títulos míticos para algunos micros provienen de esta casa, baste mencionar el Exploding Fist I y II, Hobbit, etc.

Parece que la casa Dro Soft empezará a partir de ahora a comercializar los productos de esta prestigiosa firma.

ACCOLADE CAMBIA DE DUEÑO

Tom Frixina, el antiguo Chairman de la excelente casa Accolade, ha abandonado tan famosa firma. A partir de ahora se hará cargo de la misma el segundo de a bordo, que recordaréis era ex programador de la famosa casa Activision.

Esperemos que los cambios de esta compañía no harán cambiar la calidad de los programas que ya son clásicos por el esmero que muestran en la calidad gráfica y la animación.

MASTERS DEL UNIVERSO

● US GOLD ■ ARCADE

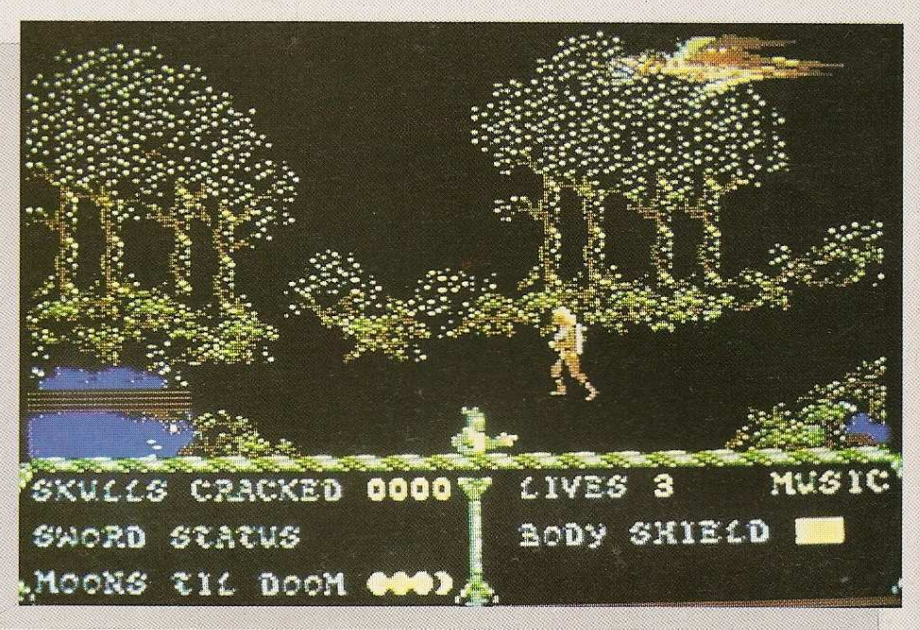
Muchas han sido las adaptaciones de películas y de máquinas recreativas transportadas al ordenador. Pero hasta ahora nunca habíamos visto un programa de ordenador, como el que os presentamos, basado en los infantiles muñecos de MASTERS DEL UNIVERSO.

El protagonista, como siempre, eres tú, investido ahora con los atuendos de GRAYSKULL, el intrépido héroe desfacedor de entuertos y en lucha constante por devolver la paz y la libertad al mágico reino de ETERNIA.

El malvado antagonista, que hará todo lo posible para destruirte, es SKELETOR, señor de una legión de infames y pérfidos esbirros y dueño del feudo de SNAKE.

Éste, robando una mágica gema, ha sembrado el caos y conquistado tu supuestamente inconquistable castillo, ambicionando dominar el universo con siniestros propósitos.

Nuestra misión será recuperar la joya y devolver el bienestar al mundo. Pero antes de lograr llevar a cabo nuestro objetivo deberemos afrontar una continua pesadilla de batallas y trampas, todo en favor del bien.



Solos ante el peligro, deberemos buscar, en primer lugar, al brujo ORKO, fiel amigo nuestro, que quedó encerrado en una torre del castillo por un *error* de los conjuros que lanzó contra las maléficas hordas que atacaban al castillo de GRAYSKULL. Una vez rescatado dicho mago, nos será de gran ayuda si queremos evitar ser alcanzados por los innumerables sortilegios y conjuros que nos enviará SKELETOR. Sólo nosotros podemos rescatarle y sólo él puede ayudarnos. Ésa será nuestra más importante misión, antes de alcanzar la libertad del territorio de ETERNIA.

Como habréis podido imaginar, este juego se desarrolla en medio del fabuloso castillo de GRAYSKULL. Salas, mazmorras, torres, almenas... serán nuestro objetivo. De paso, todo ello sembrado de trampas y demoníacos enemigos, miembros de la guardia de SKELETOR. Contra ellos la única arma de la que disponemos es la ESPADA DESTRUCTORA, que atomizará todos los poderes malignos antes de llegar a buen fin nuestra tarea.

Os advertimos, por último, de una cosa más: atención a las flechas, que nos irán quitando energía vital.



ANIMACION	7
INTERES	6
GRAFICOS	7
COLOR	7
SONIDO	6
TOTAL	33



PARK PATROL

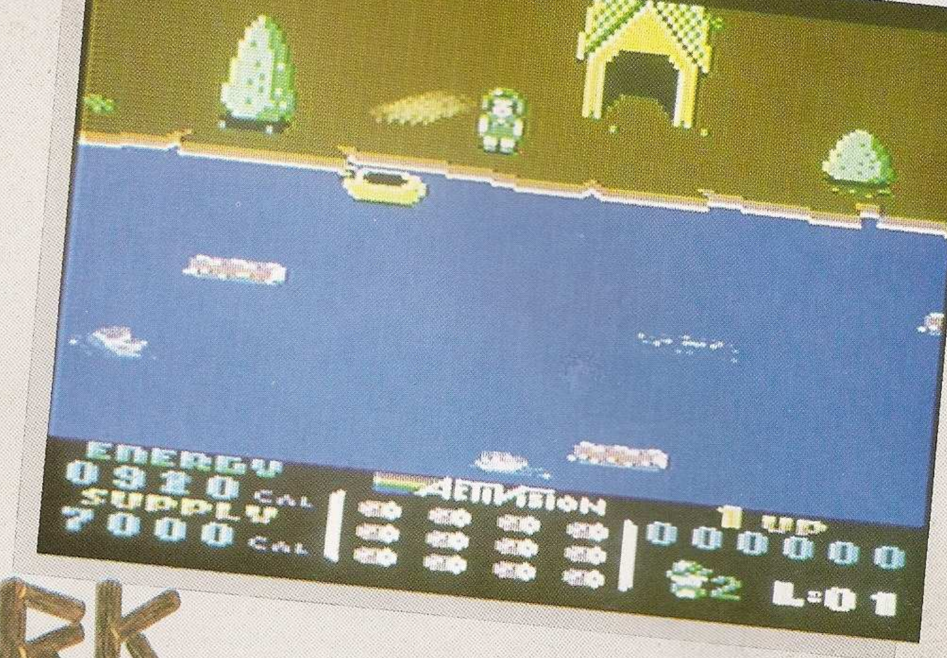
● FIREBIRD ■ ARCADE

Hay que ver qué cosas más raras han pasado en el parque PAPATOETOE.

El antiguo guardián del parque ha tenido una crisis de nervios y el parque no puede quedar sin vigilante; hace falta un sustituto. Esperamos que quieras aceptar el reto de ser el nuevo guardián del lugar.

Lo único malo del parque es que está plagado de serpientes, hipopótamos, y otras bestias salvajes que tendrás tiempo de sobra de conocer.

Tu trabajo es conservar el lugar limpio y tranquilo, evitar accidentes cuidando de que los bañistas no se extravíen, o salvar los que se hallen en peligro (en este caso aparecerá un HELP parpadeando en la parte inferior de la pantalla).



Ojo, porque el parque está repleto de rubíes de valor que te darán puntos y también ve con cuidado con las lanchas; el río que bordea PAPATOETOE es muy peligroso y plagado de misteriosas rocas que harán que quizás vuelque el barco en el que lleves al pobre bañista que hayas salvado. Recuerda que tú no sabes nadar.

Una vez más Firebird, la gran multinacional del software, nos ha presentado un juego de calidad, que combina múltiples escenarios y diferentes acciones, a fin de que no te aburras. La originalidad del juego, concebido en un parque, también merece destacarse.

Con toda seguridad muchos de vosotros os vais a encandilar frente a vuestra pantalla, intentando controlar a esos animales que, cuando los veis en el zoo de vuestra localidad tan solo parecen seres somnolientos.

ANIMACIÓN	8
INTERES	6
GRAFICOS	6
COLOR	7
SONIDO	6
TOTAL	33

DESTROYER

● EPYX ■ JUEGO ▲ DISCO

Para jugar con este programa debes tener en cuenta que el Destructor que comandas desplaza 3.050 toneladas, dispone de 16.000 galones de agua potable y sus 60.000 caballos te permitirán desplazarlo a 36 nudos marinos de velocidad si la mar es lo suficiente mansa, y con una autonomía de crucero de unas 6.000 millas.
Además disponemos de un buen arsenal bélico, del que destacan: 5 cañones de 50 mm, 10 antiaéreos, 7 cañones de 20 mm, 10 torpedos del 21, 2 raíles de ejectores de cargas de profundidad dotados de 10 lanzaderas. Todo esto e incluso más te será necesario para poder llevar a cabo con alguna probabilidad de éxito la misión que te acaba de encomendar la marina americana en un momento: destruir el máximo de barcos, submarinos, transportes, etc., del enemigo.
Efectivamente, las posibles misiones que te podrán ser encomendadas, son desde la caza de un submarino hasta la participación en un bloqueo naval, pasando por la escolta de un convoy, o el rescate de una goleta. Éste es quizás uno de los pocos juegos de Epyx (con el Gi-JOE que todos conocéis) que solamente saldrá en formato disco, ya que continuamente recurre a cargar pantallas desde el disco, que dicho sea de paso, utiliza los famosos ficheros VORPAL de los que hablamos en un número anterior.
Son en total 15 las pantallas que se obtienen tecleando un código de dos dígitos en una línea de estado que aparece en la parte inferior de la pantalla, así obtendrás la vista del radar, o bien, una preciosa pantalla de estado de la nave que te será de ayuda para remendar situaciones de peligro.



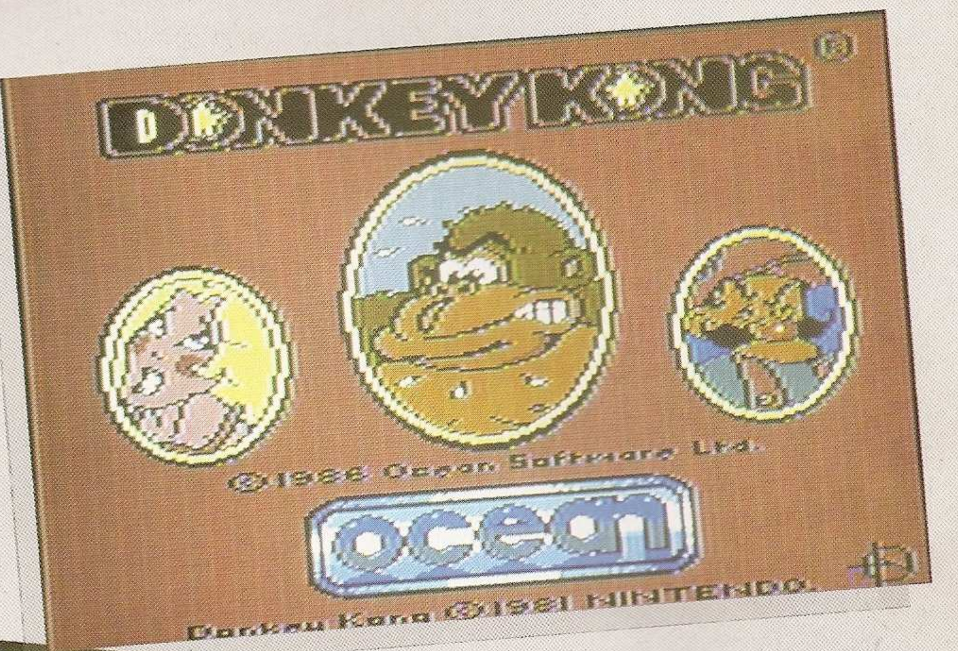
Según la misión, las pantallas por las que puedas optar van cambiando, verás como desde la pantalla del puente puedes observar cómo se van acercando los aviones y cómo vas acertando con tu batería antiaérea.
El grafismo es realmente excelente, y la animación, tal como viene siendo tradición en Epyx, se puede calificar, sin exageración, como soberbia. Una vez más, esta firma de software no ha dejado de sorprendernos con esta supernovedad de la que ofrecimos oportunamente una primicia en la sección SOFTACTUALIDAD.

ANIMACION	9
INTERES	9
GRAFICOS	10
COLOR	10
SONIDO	7
TOTAL	45



KONG

● OCEAN ■ ARCADE



Vale la pena preguntarse por qué se ha hablado tan poco en los medios especializados en videojuegos, sobre uno tan conocido como este que comentamos.

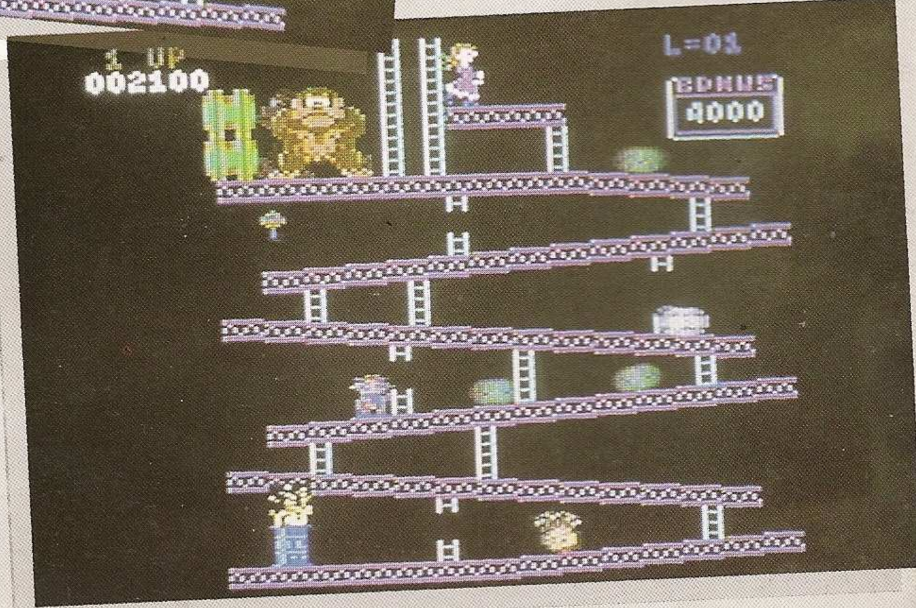
Kong, el protagonista, es un juguetón gorila que se ha encariñado tanto de tu amiga que la ha raptado. Como eres un hombre valiente y ante todo caballeroso deberás ir a rescatarla y para ello vencer al gorila en cada una de las cinco fases de que consta este juego.

En la primera fase **Kong** te pondrá difícil y peligrosa la escalada hasta donde se encuentra tu novia.

Mientras **Kong** te tirará barriles, lo difícil es saltarlos, pero cuando haya más de uno y seas capaz de saltarlos, verás que obtendrás más puntos: 500 por dos objetos y 800 por tres o más. Aunque si lo que quieres es recoger puntos no cabe duda de que obtendrás muchos más si logras subir a toda prisa evitando coger unos martillos que verás colgados del techo y que te permitirán aplastar los barriles. La segunda fase es quizás la más difícil de todas y deberás fijarte mucho en el momento en que pasa el martillo que te tira siempre, con muy mala espina, **Kong**. En la tercera y cuarta fase todo se complica con la aparición de unas llamas que parecen vivas y muchas veces te dejarán maltrecho.

Se trata de un juego que siempre ha sido muy vibrante desde su creación por la casa ATARI.

ANIMACION	7
INTERES	7
GRAFICOS	6
COLOR	6
SONIDO	6
TOTAL	32



SUPER HUEY II

● US GOLD ■ SIMULADOR DE VUELO

Aquí nos encontramos frente a uno de los principales simuladores de vuelo para helicópteros del mercado. Está concebido como una emulación del famoso UH2X, incorporando más de 50 comandos y ayuda para el vuelo, además de una completa serie de opciones que más adelante vamos a describir detalladamente.
Sin duda, el elemento distintivo de este programa es la diversidad de misiones, tanto suicidas como de rescate en zonas peligrosas. En las misiones suicidas deberás impedir que un loco y peligroso terrorista logre destruir las bases de una determinada área de la que se te ha encargado la protección. Las misiones de rescate tienen como escenario exóticos lugares que van desde el Polo Norte hasta una plataforma petrolífera en medio del Pacífico. La verdad es que éste es un helicóptero muy versátil y esto hace que sean muchos los mandos que debemos controlar y los indicadores de los que debemos estar atentos. Pero en los momentos de mayor dificultad podemos hacer uso del piloto automático para concentrar nuestros esfuerzos en tareas como la extinción de incendios.
Bien seguro que los aficionados a los simuladores de vuelo sabrán reconocer la calidad de este programa. Además podrán gozar de toda la información necesaria para pilotaje del helicóptero gracias a un completo manual del que se nos hace entrega al comprar el programa.

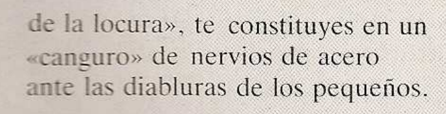
ANIMACION	8
INTERES	6
GRAFICOS	6
COLOR	6
SONIDO	5
TOTAL	31

COMMODORE 64/128

MAD NURSE

● FIREBIRD ■ JUEGO

El parvulario se ha convertido en un manicomio de la noche a la mañana. Tú, como *canguro* especializado, deberás cuidar, mientras no pierdas la paciencia y la energía, de tres plantas repletas de tiernos y *angelicales* infantes.
Los ingenuos querubines intentarán hacerte el trabajo imposible por medio de *inocentes* travesuras tales como: salirse de la cuna, ingerir biberones sin prescripción facultativa o, en el mejor de los casos, meterse en los ascensores para que tu tarea resulte más insoportable. Sus salidas en grupo harán, en más de una ocasión, que prefieras graves amarguras. Tus únicas armas para contrarrestar tan infernal ofensiva serán unas botellas de gas tranquilizante y un ascensor *prêt-à-porter* pa a cambiar de plantas. Conforme los vas metiendo en sus cunas verás cómo van apareciendo cada vez más y más legiones de incansables niñitos. La locura se irá apoderando de ti. Practicando de manera perseverante, ya estarás en condiciones de apreciar por ti mismo si, en lugar de convertirte en una niñera «presa de la locura», te constituyes en un «canguro» de nervios de acero ante las diabluras de los pequeños.





ANIMACION	9
INTERES	7
GRAFICOS	6
COLOR	6
SONIDO	7
TOTAL	35

POD

● MASTERTRONIC ■ ARCADE

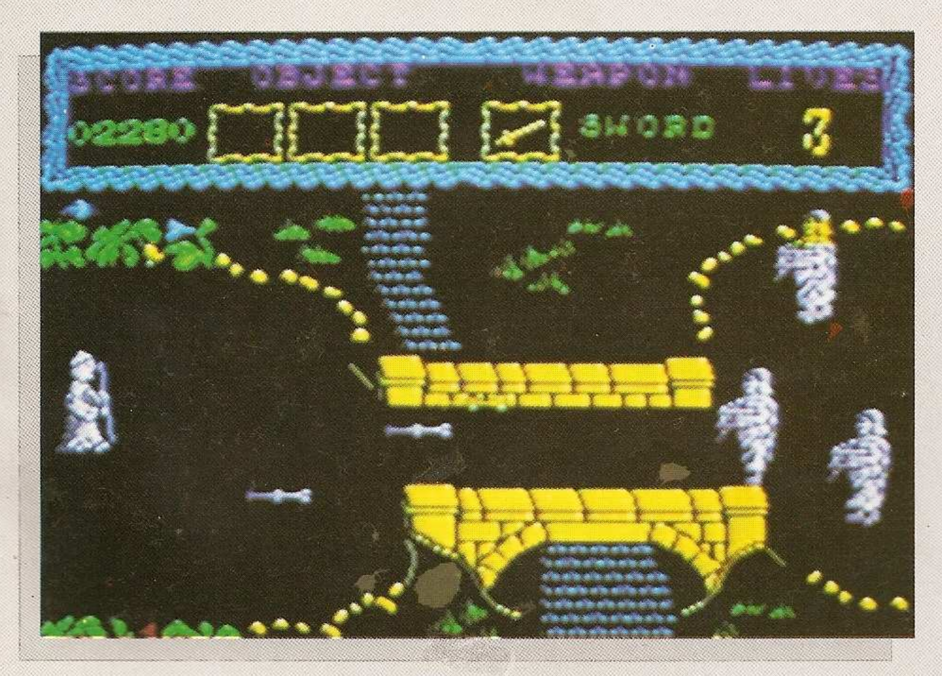
Será éste un programa que hará las delicias de los amantes del manejo del joystick: vagaremos, a gran velocidad, por un escenario repleto de hostiles hordas de atacantes, disparando sin cesar en el intento de acabar, al precio que sea, con ellas. Tiros cruzados, disparos megatónicos nos harán el trabajo más difícil de lo que inicialmente pensábamos. Los gráficos, dentro de la temática y del planteamiento, son bastante buenos. El movimiento e hipervelocidad, alucinantes. Como dato ilustrativo, te informamos que *pod*, en castellano, significa «manada».



ANIMACION	6
INTERES	7
GRAFICOS	6
COLOR	6
SONIDO	7
TOTAL	32

CURSE OF SHERWOOD

● MASTERTRONIC ■ ARCADE



Robín de los Bosques debe expulsar de la foresta al malvado Cult. Pero realmente hay poco que hacer contra tan pérfido personaje; lo único y primordial es encontrar un exorcismo que será el que logre *evaporar* a tan lamentable individuo. Los gráficos recuerdan bastante al programa ROBIN DE SHERWOOD: el *scroll* del protagonista está muy bien realizado sobre un fondo negro que representa al bosque. Las dificultades que atravesaremos serán numerosas y difíciles: el bosque está plagado de peligros, entre ellos las infames trampas de los sicarios a sueldo de CULT que disparan sus certeros dardos antes de preguntar quien eres. Suerte y atención a los puentes, esconden innumerables trampas.

ANIMACION	7
INTERES	7
GRAFICOS	6
COLOR	6
SONIDO	6
TOTAL	32

EL ZOCO

Cambio programas en disco para C-64. Salvador Pou. P.° Dr. Maragas n.° 204. Barberà del Vallès. (08210). Barcelona.

Vendo C-64, Datassete C2N, 400 programas de todo tipo, 2 joystick, revistas y libros sobre Commodore. Todo por 35.000 pts. Narciso Quintana Varo. Plaza Buigas, 2, 5.° 1.ª. Cerdanyola. Barcelona. Tl: 6916646.

Compro C-64, cassette y joystick; interface midi y programas Midicomposer de MICROMUSIC o similar. Gabriel Díaz. Apdo. 93045. Barcelona 08080. Tl: (93) 2179080.

Intercambio programas de todo tipo en cinta para el C-64. Dispongo de una amplia variedad de programas. David Ramos Isús. C/ Nuria, 79, 1.° 1.ª Montcada i Reixac. 08110 Barcelona. Tl: 5641203 por la tarde.

Vendo cartucho SIMON'S BASIC con instrucciones. Tl: (93) 3356333 a partir de las 21.20 h. Amancio Pérez. Barcelona.

Intercambio de programas para el C-64. Últimas novedades. Jorge Peña. Avd. Paralelo, 114, 5.°, 2.ª, esc. dcha. Barcelona 08015. Tl: 3299281.

Vendo C-64, datassette, manual del usuario (castellano), libros especializados (2), revistas, juegos muy buenos y programas de utilidades. Todo en perfecto estado. Precio a convenir. Eduardo Dorgambide. C/ Gasset, 23. Puebla del Caramiñal. La Coruña. Tl: (981) 830410.

Cambiamos programas y juegos para el C-64 (preferentemente de calidad). Interesados ponerse en contacto con Manel Soriano. C/ Castellar, 82. Sabadell. Barcelona Tl: (93) 7160730.

Vendo impresora SEIKOSHA GP-100VC especial commodore. Como nueva. Por sólo 25.000 pts. Regalo además 300 juegos en cinta para el C-64. Llamar mañanas de 8 a 3. Tl: (93) 3023200 ext. 280. Gabriel Reina.

Me gustaría forma un Club de amigos del C-64 para cambiar ideas para la compra-venta y el intercambio de juegos. Los chicos/as tienen que tener una edad entre los nueve y los trece años. A ser posible que vivan cerca de Sant Andreu. Juan Antonio Fdez. C/ Condesa Pardo Bazán, 15, Atc. 3.ª. Barcelona 08027. Tl: 3498113.

Cambio todo tipo de programas para el C-64. También me interesan pequeños trucos. Luis Tapia Jiménez. C/ NR Los Príncipes. La Fontanilla, 3. 41009 Sevilla.

Intercambio utilidades en disco para el C-64, C-128 y CP. M. Antonio González. C/ Lorena, 65-67 7.°, 2.ª. 08031 Barcelona. Tl: (93) 3592300.

Vendo material para el C-64, 128. Hay revistas, libros y juegos. Si os interesa dirigiros a: Oscar Fdez. Orallo. Pza. La Fortaleza, 11, 4.°. 24400 Ponferrada. León. Tl: (987) 418573.

Ofrezco 10 juegos a elegir por el programa MAGIC DESK I, en cinta. David Guerrero Díaz. C/ Emilio Santacana, 5. Algeciras. Cádiz.

Cambio/vendo juegos para el C-64. Poseo últimas novedades. Llamar Tl: (968) 590357 o escribir a: Tomás Fuentes Mández. Avda. Constitución, 63. Mazarrón (30870) Murcia.

Cambio programas en cinta y disco para el C-64 y 128. Xavi Sanahuja Anguera. C/Jurats, 5, 4.° 43205 Reus.

Cambio/vendo programas para el C-64 en cinta. Poseo las últimas novedades, precios muy asequibles. Luis San José Fernández. C/ General Shelly, 25, 3.° D 47013 Valladolid. Tl: (983) 277371.

Club Intercommodore con más de 50 socios espera que tú también te apuntes, para ser más. C. I. C. C/ Algorta, 9. Buzón, 9. 28019. Madrid.

Vendo C-64 (nuevo diseño) junto con Datasette 1530, ambos con embalajes originales, manuales en español e inglés y en perfecto estado de conservación. Por 55.000 pts. Juan Muñoz Falcó. Avd. Suecia, 4, 30. Valencia. 46010. Tl: (96) 3699571.

Intercambio programas para el C-64, trucos, experiencias con C-128. Los programas preferentemente en turbo. Hilario García Ostos. C/ General Queipo de Llano, 14. Peñarroya Pueblonuevo. Córdoba. Tl: (957) 560231.

Intercambiamos juegos, utilidades o experiencias. Club de Usuarios de Commodore 64. Tenemos últimas novedades. Felipe Carrero. Cuarteles, 142 4.° B. Tl: 8920746. Aranjuez. Madrid.

Intercambio juegos en cinta para C-64. Poseo más de 500, entre ellos parte de las últimas novedades. Luis Poves Valencia. C/ Prado, 12, 12.° 2.ª L'Hospitalet de Llobregat. 08907 Barcelona. Tl: (93) 3355405.

Cambio juegos-utilidades para el C-64/C-128. Sólo discos. Enviar lista a: Javier Covarrubias Morales. C/ Portalegre, 27 2.° dcha. 28019 Madrid.

Intercambio programas para el C-64. Poseo más de 200 programas de gran calidad. Interesados enviar lista. Pedro Navarro. P.° Almogávares, 28. Sabadell. Barcelona.

Club de amigos para el C-64 intercambia programas en cassette. Tenemos más de 100. Creación de una revista para los socios con mapas, cargadores, trucos. José. Apartado de Correos, 195. Lérida.

Vendo para el C-64 los juegos: Fist y Beach Heat II por mil pts. También los cambio por otros que me interesen: Preguntar por Kelius. Tl: (93) 2396149.

Desearía cambiar juegos o aplicaciones para el C-64, noches de 8.30 a 10 h. Tl: (93) 6640456. Barcelona.

CINCO MINUTOS ANTES DE COMPRAR UN JUEGO A 875 Ptas.

ECHALE UN VISTAZO A ESTOS JUEGOS DE 875 Ptas.



875 Ptas.

VERSION CASSETTE

COBRA SOFTWARE

SÍGUENOS EL JUEGO

ZAFIRO SOFTWARE DIVISION Paseo de la Castellana, 141. 28046 Madrid
Tel. 459 30 04. Tel. Barna. 209 33 65. Telex: 22690 ZAFIR E